UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Interpercepção: Uma Arquitetura de Software para Compartilhamento de Ambientes Virtuais com Interfaces Gráficas Distintas

Rummenigge Rudson Dantas

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Número de ordem PPgEE: M000
Natal, RN, abril de 2008

Divisão de Serviços Técnicos

Catalogação da publicação na fonte. UFRN / Biblioteca Central Zila Mamede

Dantas, Rummenigge Rudson.

Interpercepção: uma arquitetura de software para compartilhamento de ambientes virtuais com interfaces gráficas distintas / Rummenigge Rudson Dantas - Natal, RN, 2008

83 p.

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves

Disertação (mestrado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.

1. Redação técnica - Disertação. I. Gonçalves, Luiz Marcos Garcia. II. Título.

RN/UF/BCZM CDU 004.932(043.2)

Interpercepção: Uma Arquitetura de Software para Compartilhamento de Ambientes Virtuais com Interfaces Gráficas Distintas

Rummenigge Rudson Dantas

Dissertação de Mestrado aprovada em 15 de fevereiro de 2008 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Marcos Garcia Gonçalves (orientador) . . . . . DCA/UFRN

Prof. Dr. Jauvane Cavalcante de Oliveira . . . . . LNCC/MCT

Prof. Dr. Luiz Affonso Henderson Guedes de Oliveira . . . . . DCA/UFRN

A minha grande família e aos meus queridos amigos.

Agradecimentos

Ao meu orientador, professor Luiz Marcos Garcia Gonçalves, sou grato pela orientação.

Aos amigos, especialmente a família Natalnet, pelo apoio durante a realização deste trabalho.

Aos demais colegas de pós-graduação, pelas críticas e sugestões.

À minha família, especialmente meus pais, pela paciência durante esta jornada.

À CAPES, pelo apoio financeiro.

Resumo

Neste trabalho, é apresentado o paradigma Interperceptivo, uma nova abordagem que inclui um conjunto de regras e uma arquitetura de software que une usuários de interfaces diferentes no mesmo ambiente virtual. O sistema detecta os recursos do usuário e realiza transformações nos dados para sua visualização nas interfaces 3D, 2D e textual (1D). Isto permite que qualquer usuário conecte, acesse todas as informações, e troque informações com os outros usuários de um modo flexível, sem necessitar de alterações no hardware e sofware. Como resultados são apresentados dois ambientes virtuais construídos segundo este paradigma.

Palavras-chave: Ambiemtes Virtuais, Sistemas Multiusuários.

Abstract

In this work, we propose the Interperception paradigm, a new approach that includes a set of rules and a software architecture for merge users from different interfaces in the same virtual environment. The system detects the user resources and provide transformations on the data in order to allow its visualization in 3D, 2D and textual (1D) interfaces. This allows any user to connect, access information, and exchange information with other users in a feasible way, without needs of changing hardware or software. As results are presented two virtual environments builded acording this paradigm.

Keywords: Virtual Environments, Multi-user Systems.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

API	Acrônimo em inglês para *Aplication Programing Interface* que define um conjunto de rotinas e padrões estabelecidos por um software para utilização de suas funcionalidades por programas aplicativos.
BBS	Acrônimo em inglês para *Buletin Board System*, que define um *software* que permite a conexão remota de dois ou mais computadores através da linha telefônica. É portanto, um antecessor da *internet*.
Engine	Motor gráfico. Nome dado a softwares que possuem um conjunto de ferramentas para o desenvolvimento de aplicações gráficas
HMD	Acrônimo em inglês para *Head-Mounted Displays*. É dispositivo de realidade virtual que consiste em um capacete com visor acoplado. Este visor possui um par de monitores de cristal líquido que exibem imagens geradas por computador e visam conceder ao seu usuário uma sensação de imersão em um ambiente virtual.)
IHC	Acrônimo em português para Interação Humano Computador. Trata-se de uma sub-área da computação que estudo técnicas e metodologias para melhorar a interação do homem com a máquina (neste caso, os computadores)
LAN	Acrônimo em inglês para *Local Area Network*, que em português significa Área de Rede Local. Define uma rede local, para uma gama pequena de usuários
midleware	Camada de software usada como ponte entre o software e o hardware de algum computador.
MMOG	Acrônimo em inglês para *Massive Multiplayer Online Game*, que define jogos *on-line* para uma quantidade massiva de usuários, geralmente da ordem de centenas de usuários.
MMORPG	Acrônimo de *Massive Multi-player On-line RPG* e define os jogos de RPG multi-usuários para uma quantidade massiva de jogadores, da ordem de centenas.
MUD	Acrônimo em inglês para *Multi-user Dungeon*, que define um sistema multiusuário para simulação de jogos de **RPG**. Um MUD é executado via **BBS** ou pela *Internet*.

MVC — Acrônimo em inglês para *Model-View-Controller*. Padrão de design que separa o código de um sistema em três porções: uma porção que armazena as informações vitais ao funcionamento do sistema (*Model*); outra dedicada a interface gráfica do sistema (*View*); e uma porção dedicada ao controle geral do sistema (*Controller*).

NPC — Acrônimo de *Non-Player Character*. Definição da aréa de jogos para os personagens que são controlados pelo computador, por técnicas de Inteligência Artificial.

RPG — Acrônimo em inglês para *Role-Playing Games*, ou jogos de interpretação de papéis em português. Define um tipo de jogo, no qual seus participantes interpretam personagens que participam de uma narrativa fantasiosa criada por um dos participantes do jogo. Nas versões eletrônicas deste jogo, as narrativas já são estabelecidas pelo criador do jogo.

Capítulo 1

Introdução

Antes do surgimento da *Internet*, a interligação remota entre dois ou mais computadores era feita através de um **BBS** (*Bouletin Board System*). Esse sistema de conexão permitia que seus usuários pudessem trocar informações através da linha telefônica. Em geral, essas informações eram mensagens eletrônicas (*e-mail*), mas podiam ser qualquer outro tipo de dado disponível.

No final da década de 70, Roy Trubshaw e Richard Bartle, dois pesquisadores da Universidade de Essex, utilizaram um **BBS** para executar um jogo onde vários usuários conectados interagiam através de uma *interface* de visualização textual. O jogo combinava elementos de **RPG** (*Role-Playing Games*) e interação social através de salas de bate-papo (*chat*). Ele foi batizado de Multi-user Dungeon [Bartle 1990] em referência ao jogo de **RPG** Dungeons and Dragons [Gygax 1987] porém, ficou mais conhecido pelo acrônimo **MUD**. A figura 1.1 ilustra a *interface* de visualização de um cliente **MUD**. Como pode ser obeservado pela figura, a *interface* de visualização deste tipo de aplicação é puramente textual. Mesmo com uma *interface* de visualização textual, é possível considerar o **MUD** como o marco inicial para criação de ambientes virtuais multiusuários *on-line*.

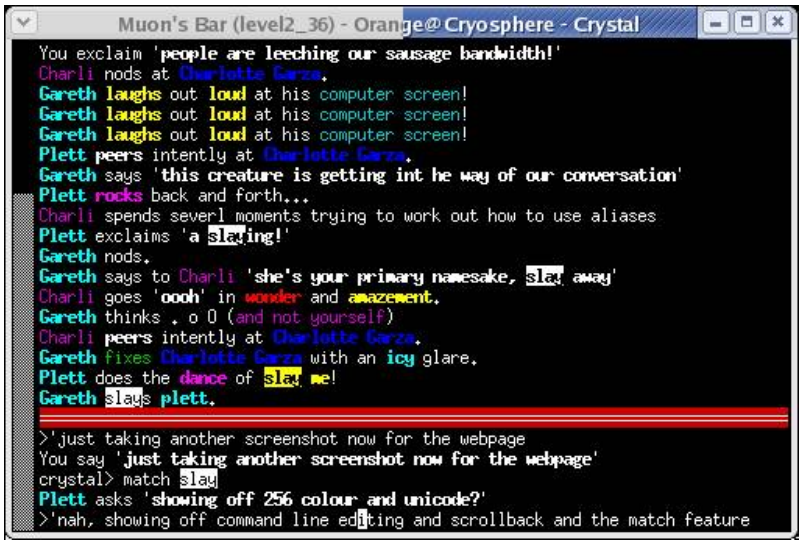

Figura 1.1: Imagem da apliacação cliente de um **MUD**

Na década seguinte, surgiram os primeiros ambientes com *interface* de visualização 2D. Nesse conjunto de ambientes, merece destaque o *Habitat* [Morningstar e Farmer 1991] desenvolvido por Chip Morningstar e Randy Farmer para a fábrica de jogos Lucas-Film Games. A *interface* de visualização 2D do *Habitat* permite aos usuários visualizar o ambiente numa representação gráfica, em contraste com a representação descritiva de qualquer **MUD**. Os próprios usuários do *Habitat* possuem uma representação gráfica e animada, que, na época do *Habitat*, foi denominada de *avatar*. A figura 1.2 mostra uma imagem do *Habitat*.

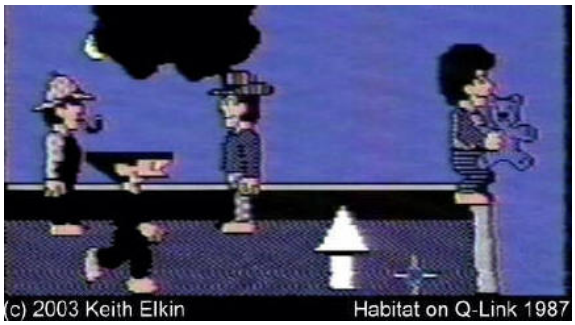

Figura 1.2: Interface do *Habitat*

A década de 90 foi marcada pelo apogeu da *Internet* e o lento desaparecimento dos **BBS**'s. Também nesta década, surgiram os primeiros ambientes *on-line* (entenda-se também por ambientes virtuais multiusuários) com *interface* de visualização 3D. Nessa geração, o principal marco no âmbito dos ambientes *on-line* foi o *Active Worlds* [ActiveWorlds 2007]. O *Active Worlds* foi originalmente concebido para ser um navegador 3D da *Internet*. Porém, foi convertido em um ambiente virtual *on-line*. Na figura 1.3, apresentamos um imagem da *interface* de visualização do *Active Worlds*.

Figura 1.3: interface de visualização do *ActiveWorlds*

Atualmente, existem ambientes virtuais com esses três tipos de *interface* de visualização (textual, 2D e 3D). O surgimento de ambientes com *interfaces* gráficas mais

elaboradas e com mais dimensões não extinguiu os ambientes com *interface* de visualização mais simples. Ainda hoje, é possível encontrar **MUD**'s funcionando na *Internet* [MUD 2007]. Do mesmo modo, continuam surgindo ambientes com interface de visualização 2D, porém, com suporte a uma quantidade massiva de jogadores, como o Ragnarok [Gravity 2007].

Esta situação pode ser explicada pelo fato de que *interfaces* mais complexas exigem mais requisitos de *hardware* e nem todos os usuários possuem tais requisitos, principalmente nos países emergentes. Além disto, não é consenso que uma *interface* 3D seja sempre mais apropriada do que uma 2D. Por isso, esses usuários buscam ambientes com *interfaces* de visualização mais simples. Além disto, nem todos os usuários conseguem ou gostam de lidar com a quantidade de informação visual que é apresentada na maioria das *intrefaces* 3D. Por isso alguns preferem utilizar ambientes com *interface* de visualização 2D ao invés de uma interface de visualização 3D.

Partindo-se do pressuposto de que existem usuários com diferentes requisitos de *hardware* e software e de que existem pelo menos três tipos de interface para visualização (e conseqüentemente interação) com ambientes virtuais, propomos neste trabalho uma arquitetura que integre os usuários que estão conectados ao ambiente virtual *on-line*, mesmo que a interface que todos esses usuários utilizam para visualizar o ambiente não seja a mesma. Utilizando um ambiente projetado com base nesta arquitetura, um usuário poderá optar por qualquer tipo de interface de visualização que esteja disponível, tendo a garantia de que será capaz de visualizar os demais usuários que estão conectados ao mesmo tempo, independente de qual interface de visualização os mesmos escolheram para acessar o ambiente.

Num ambiente construído segundo este paradigma, os usuários percebem o ambiente de acordo com a sua interface de visualização, independente da interface de visualização alheia. Deste modo, dizemos que os usuários num ambiente assim, percebem uns aos outros de forma transparente. Ao processo de integração de ambientes com *interfaces* de visualização de dimensões distintas, fazendo uma analogia da interface de visualização textual como sendo uma interface de visualização 1D, denominamos de *interdimensionalidade*. A técnica utilizada para permitir que os usuários percebam uns aos outros de modo transparente denominamos de *interpercepção*.

1.1 Motivação

A motivação ao desenvolvimento deste trabalho nasceu da existência diversos perfis de usuários que utilizam ambientes virtuais multiusuários através da *Internet*. Podemos encontrar usuários que possuem computadores com alto poder de processamento conectados a uma rede de alta velocidade, como também podemos ter clientes utilizando computadores antigos, com uma configuração de *hardware* e *software* obsoletas, conectados através de uma conexão discada. Um ambiente virtual que disponibilize diferentes *interfaces* de visualização para que seus usuários o acessarem irá garantir uma maior acessibilidade, principalmente para os usuários com poucos requisitos. Usuários com equipamentos de baixo desempenho, que não conseguem acessar o ambiente através de uma *interface* de visualização 3D, podem optar por *interfaces* de visualização menos exigentes, como

a *interface* de visualização 2D. Já usuários com deficiência visual poderiam, através das técnicas de síntese e reconhecimento de voz, utilizar o ambiente através da *interface* de visualização textual.

O ponto de partida para o desenvolvimento das teorias abordadas neste trabalho foi o sistema *PercepCom* [Dantas et al. 2005], desenvolvido por nós em trabalhos anteriores. Este sistema atua na criação de ambientes virtuais multiusuários *on-line*, sendo descrito sucintamente a seguir.

1.1.1 O sistema *PercepCom*

O *PercepCom* surgiu como um componente gráfico independente de plataforma para a criação de ambientes virtuais tridimensionais. O projeto e a implementação do *PercepCom* foram motivados pela necessidade de criação de um sistema portável, com suporte a vários usuários em um ambiente virtual colaborativo através da web. Ele foi construído com base na arquitetura cliente-servidor e foi implementado com a **API** (*Aplication Programing Interface*) Java3D. O *PercepCom* se mostrou satisfatório com relação à interatividade entre os usuários e o ambiente. Contudo, testes realizados com o ambiente 3D demonstraram que seu acesso através de máquinas com poucos recursos de vídeo era lento, tanto para navegação pelo ambiente quanto na atualização dos gráficos da *interface* de visualização. Mediante este problema de baixo desempenho do ambiente 3D em computadores de baixo potencial gráfico, foi idealizado um ambiente que utilizasse uma *interface* de visualização 2D ao invés da 3D. Assim surgiu o *PercepCom*2D [Paulino et al. 2006] e a versão anterior foi renomeado de *PercepCom*3D.

Mesmo que uma gama alta de clientes seja alcançada com estas duas versões do *PercepCom*, foi ainda proposta uma terceira *interface* de visualização, puramente textual, inspirada nos **MUD**'s. Esta última *interface* de visualização recebeu o nome de *PercepCom*1D. Esta versão textual foi também proposta para integração de usuários portadores de deficiência visual e conta com ferramentas de síntese e reconhecimento de voz. A figura 1.4 mostra as três *interfaces* de visualização do *PercepCom*.

1.2 Contribuições

A principal contribuição deste trabalho é o desenvolvimento da arquitetura de um sistema que permita a integração das diferentes versões de um ambiente virtual, formando um único ambiente que é compartilhado por todos os usuários, onde estas versões possuem *interfaces* de visualização distintas para visualização e interação. Neste ambiente compartilhado, todos os usuários conectados podem interagir mutuamente. Além desta contribuição principal, as seguintes contribuições (secundárias) também são frutos do presente trabalho:

- Desenvolvimento de um servidor unificado capaz de receber e tratar mensagens oriundas das diferentes versões de um mesmo ambiente virtual;
- Criação e implementação de algoritmos para conversão de espaços entre as diferentes *interfaces* de visualização (com dimensões espaciais distintas) do ambiente;

Figura 1.4: Interfaces do Sistema *PercepCom*

- Desenvolvimento de um componente de *software* capaz de converter mensagens de controle oriundas de aplicações cliente com *interfaces* de visualização distintas;

1.3 Metodologia

A execução desse trabalho considerou o paradigma de engenharia de *software* denominado prototipagem, baseando-se no desenvolvimento de um protótipo inicial, a partir do qual atingiu-se, por melhorias constantes, o produto final. Este modelo de desenvolvimento é também chamado de paradigma evolutivo. Para modelagem arquitetural do sistema apresentado neste trabalho, usamos o padrão de design (*design pattern*) [Gamma et al. 1999] **MVC** (*Model - View - Controller*) e o padrão de design *Strategy*. Como ferramenta de modelagem, foi utilizado o JUDE [JUDE 2007] e as notações UML. Como ferramenta de edição e atualização de código foi utilizado o Eclipse [Eclipse 2007], que permite facilmente o controle de versões do sistema (usando a técnica CVS).

1.4 Estrutura deste trabalho

Após a introdução, aqui finalizada, apresentaremos no Capítulo 2 um embasamento teórico sobre os conceitos que são vitais para o entendimento e elaboração deste trabalho.

No capítulo 3, são analisados alguns trabalhos que apresentam uma abordagem semelhante à nossa. A formalização do problema abordado neste trabalho, ou seja, a integração de ambientes virtuais com *interfaces* de visualização distintas, é tratada no Capítulo 4.

A *interpercepção*, nossa abordagem para integrar os ambientes, é descrita e detalhada no Capítulo 5.

No Capítulo 6 apresentamos os resultados dos testes e experimentos realizados para validar nossa proposta.

Finalmente, no Capítulo 7, apresentamos nossa conclusão, incluindo possibilidades de trabalhos futuros relacionados diretamente com o nosso tema.

Capítulo 2

Embasamento Teórico

O trabalho aqui proposto baseia-se em uma série de conceitos fundamentais que estão agrupados em três tópicos: Ambientes Virtuais, Sistemas Distribuídos e Interação Humano-Computador. Esses tópicos e outros sub-tópicos relacionados são discutidos a seguir.

2.1 Ambientes Virtuais

O conceito de ambientes virtuais foi introduzido por Ivan Sutherland [Sutherland 1965] em 1965 através de suas teorias sobre a inserção de pessoas em ambientes gerados por computador. Ele demonstrou suas teorias através da projeção de modelos tridimensionais em *wireframes*, utilizando um par de telas montadas em um tipo de capacete, chamados de *Head-Mounted Displays* ou **HMD**'s. O **HMD** havia sido patenteado em 1960 pela Philco [Elllis 1991], mas foi o dispositivo desenvolvido por Sutherland em 1968 [Sutherland 1968] o primeiro a usar imagens geradas por computação gráfica.

Outra importante contribuição para consolidação do termo ambiente virtual foi dada por Steve Ellis [Ellis 1994]. Ele definiu o termo *virtualização* como sendo o processo pelo qual um espectador humano interpreta uma impressão sensorial que representa um objeto em um ambiente diferente daquele no qual a impressão se originou fisicamente. Ellis dividiu o processo de *virtualização* em três níveis:

- Espaço virtual - espaço onde será projetada a imagem virtual;
- Imagem virtual - percepção de profundidade dos objetos;
- Ambiente virtual - o usuário torna-se parte do mundo virtual.

Unindo as teorias destes dois autores, podemos definir um Ambiente Virtual como sendo um ambiente sintético composto por uma modelagem gráfica gerada por computador e por um conjunto de elementos que estimulam os sentidos do usuário, simulando a sua presença neste ambiente.

Atrelado ao conceito de ambiente virtual surgiram outros conceitos tais como comunidades virtuais, mundos virtuais, ambientes virtuais multiusuários, entre outros. A seguir apresentamos as definições sobre esses conceitos.

2.1.1 Comunidades Virtuais

Comunidades virtuais baseiam-se no conceito mais geral de comunidade. O termo virtual é acrescido para diferenciar sua natureza, ou seu campo de atuação, que, nesse caso, é o espaço virtual. Portanto uma comunidade virtual nada mais é do que um grupo de pessoas interligadas por interesses, metas ou valores comuns, dentro de um espaço virtual. A *Internet* tem sido atualmente este espaço virtual, onde surgem a cada dia novas comunidades virtuais, dentre as quais o Orkut [Google 2007] tem se destacado gradativamente. A figura 2.1 mostra a tela inicial do Orkut.

Apesar de difundidas, as comunidades virtuais ainda não contemplam aquilo que geralmente esperamos de um ambiente virtual: a modelagem topológica que nos forneça uma noção de espaço. É neste ponto que introduzimos o conceito de mundo virtual.

Figura 2.1: Interface do Orkut

2.1.2 Mundo Virtual

Definimos acima os termos ambientes virtuais e comunidades virtuais, porém um outro termo comum que encontramos em algumas pesquisas sobre o assunto é o termo mundo virtual (*virtual world*). Um mundo virtual pode ser entendido como um ambiente virtual compartilhado, dotado de uma representação espacial, regido por leis e grandezas físicas, principalmente o tempo, onde os seus visitantes (ou usuários, numa linguagem mais técnica) são representados e interagem mutuamente através de avatares. Esta definição demonstra que o conceito de ambiente virtual é mais abrangente que o conceito de mundo virtual, porém não é difícil encontrar estes dois termos sendo associados como se fossem sinônimos. A figura 2.2 mostra uma imagem do mundo virtual Hive7 [Seven 2007] desenvolvido com a tecnologia Ajax [Dave Crane 2006] e que é executado diretamente por browser.

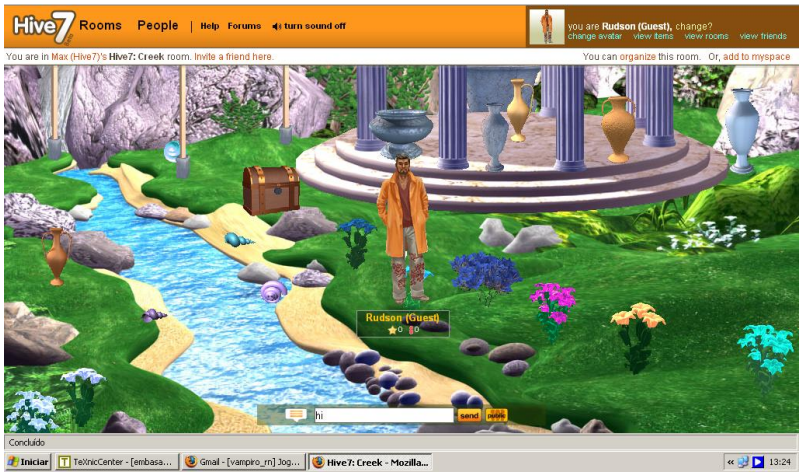

Figura 2.2: Interface do Hive7

2.1.3 Avatar

O termo *avatar* é originário da mitologia indiana. A palavra vem do sânscrito Avatāra que significa "descida" ou "encarnação". É usada nos textos mitológicos indianos para se referir à materialização de um deus na forma humana, para poder caminhar sobre a terra. No contexto da realidade virtual, o termo *avatar* é usado para designar um ícone ou qualquer outra representação visual do usuário dentro de um ambiente virtual compartilhado [Damer 1997]. O *avatar* define o modo como os usuários do ambiente virtual percebem uns aos outros. O termo foi usado pela primeira vez, com este significado, em 1985 no jogo Ultima IV. Foi adotado posteriormente pelo *Habitat* (1987) e popularizado por Neal Stephenson no seu romance Snow Crash [Stephenson 2007]. A figura 2.3 mostra exemplos de *avatares*.

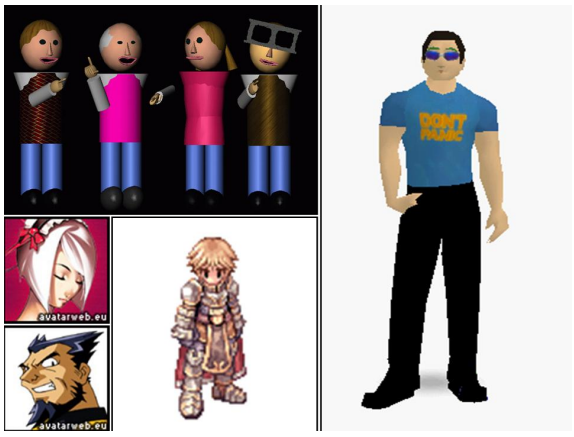

Figura 2.3: Exemplos de Avatares

2.1.4 Ambientes virtuais multiusuário

Com base nos conceitos que já apresentamos até aqui, podemos definir ambientes virtuais multiusuários como sendo ambientes caracterizados por suportar espaços sintéticos que são compartilhados por múltiplos usuários, remotamente localizados. O acesso a estes ambientes através da *Internet* é importante, porque permite que uma ampla gama de aplicações, de entretenimento à educação e medicina, seja oferecida, através de uma *interface* bastante difundida como os navegadores da *web*. A figura 2.4 apresenta uma imagem de desmonstração do River City [Dieterle e Clarke 2007], um ambiente virtual mutiusuário desenvolvido por pesquisadores na universidade de Havard. O intuito do projeto River City é servir como uma plataforma auxílo ao ensino de ciências e tecnologia para alunos do ensino médio.

Figura 2.4: Interface do River City

Outra vertente das pesquisas sobre ambientes virtuais que tem crescido bastante é a dos ambientes virtuais de grande escala (*massive*) [Greenhalgh 1999]. A adição do termo larga escala apenas amplia o conceito de ambientes virtuais multiusuário para ambientes capazes de suportar uma gama muito grande de usuários. A construção de ambientes de larga escala tem sido impulsionada principalmente pela indústria de jogos, após o surgimento dos **MMOG** (*Massive Multiplayer Online Game*). Os MMOG's nada mais são do que jogos multiusuários capazes de suportar milhares de jogadores conectados ao seu mundo virtual através da *Internet*, ou até mesmo através de uma **LAN** (*Local Area Network*).

2.1.5 Ambientes colaborativos

Ambientes Virtuais Colaborativos [Benford e Greenhalgh 2001] envolvem o uso da tecnologia de realidade virtual distribuída para apoiar o trabalho em grupo. Uma condição

necessária, mas que não chega a ser suficiente para a criação de ambientes colaborativos, é a existência de uma conexão multiusuário com um sistema de realidade virtual. Portanto, não se pode afirmar que qualquer ambiente multiusuário é também um ambiente colaborativo. A essência dos ambientes colaborativos está na construção de um sistema que forneça meios para que os seus usuários realizem atividades em conjunto. Os meios comumente utilizados para isso são as ferramentas de comunicação, tal como os aplicativos de bate-papo.

2.2 Sistemas Distribuídos

Segundo Tanenbaum [Tanenbaum e van Steen 2006], um sistema distribuído é "*uma coleção de computadores independentes que se apresentam ao usuário como um sistema único e consistente*". Num sistema como esse, o poder computacional de todas as máquinas envolvidas é somado. Este poder é então direcionado para a realização de uma tarefa colaborativa. A execução desta tarefa ocorre de forma transparente para o usuário, uma vez que para o ele todo o processamento é aparentemente feito por uma única máquina. Mesmo que o usuário tenha noção de que o processamento é feito por várias máquinas, ele ainda não seria realmente capaz de definir que parte da tarefa está sendo realizada por qual máquina, o que ainda configuraria o sistema como transparente.

Um sistema distribuído não se configura apenas por operar em várias máquinas interligadas, pois o poder computacional gerado por essa ligação também pode ser alcançado com a utilização de vários processadores trabalhando em paralelo num mesmo computador. Quando um sistema distribuído é construído desta forma, ele está seguindo uma arquitetura multiprocessadores, sendo a memória compartilhada por todos os processadores. Se existe uma memória dedicada para cada processador ele é um sistema distribuído de multicomputadores, mesmo que todos os processadores e memórias estejam numa mesma máquina ou máquinas diferentes.

A principal diferença entre sistemas com um processador e sistemas distribuídos está na comunicação entre os processos, também definido como comunicação inter-processos. Em sistemas com um processador apenas, essa comunicação é realizada através de uma estrutura de memória compartilhada. Nos sistemas distribuídos a comunicação inter-processos ocorre através de modelos (ou arquiteturas) para comunicação distribuída. O modelo de comunicação distribuída utilizado nesse trabalho é a Arquitetura Cliente-Sevidor, que definimos a seguir.

2.2.1 Arquitetura Cliente-Sevidor

Essa arquitetura define um modelo de comunicação estabelecido por dois componentes básicos: o cliente e o servidor. O cliente é um processo que atua na requisição de informações. Essa requisição é feita ao servidor. O servidor tem a tarefa de atender às requisições do cliente. Outro modo de definir o papel de cada componente da arquitetura é tratar o servidor como um prestador de serviços, o que evidência bem o seu nome. Já o cliente é alguém que busca algum tipo de serviço oferecido pelo servidor.

A figura 2.5 mostra um sistema seguindo a arquitetura cliente-servidor. Nesta figura temos apenas um cliente e um servidor, contudo é comum a existência de um conjunto de clientes buscando as informações no servidor. Nesta figura o cliente e o servidor utilizam um protocolo simples de Requisição (enviada pelo cliente) e Resposta (enviada pelo servidor).

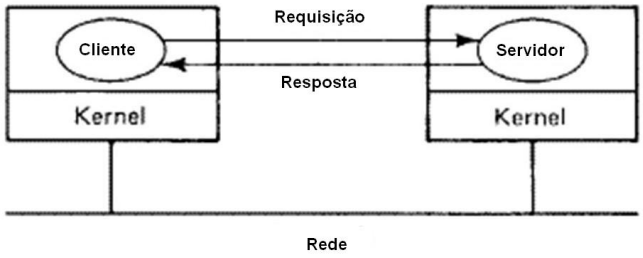

Figura 2.5: Arquitetura Cliente-Servidor

Exemplos de serviços que utilizam tal arquitetura são facilmente encontrados, como por exemplo, alguns serviços de Bate-Papo, alguns Jogos *On-line*, HTTP, DNS, entre outros. Nesses serviços há mais de um cliente e possivelmente mais de um servidor também, obrigando a uma estruturação do sistema no que diz respeito a como os clientes e servidores se relacionam.

Existem outros modelos de comunicação distribuída como a Peer-to-Peer [Schollmeier 2002]. Ao invés de explorarmos outros modelos de comunicação distribuída, tratamos doravante do conceito de comunicação em grupos.

2.2.2 Comunicação em Grupos

Segundo Tanenbaum [Tanenbaum e van Steen 2001], um grupo é "*uma coleção de processos que agem juntos em algum sistema ou de alguma forma específica*". Em um sistema distribuído, o número de processos envolvidos numa comunicação pode variar desde uma simples troca de dados entre dois processos até o compartilhamento de dados entre milhares de processos. Portanto, é necessária a existência de uma estrutura de coordenação entre esses vários processos. Para realizar tal coordenação é vital que seja desenvolvido um modelo para comunicação em grupo, o que permite desta forma a criação de grupos de processos que cooperem para fornecer um serviço. Basicamente, a implementação dessa comunicação pode ser feita através de três modelos: *unicast*, *multicast* e *broadcast*.

No modelo *unicast* a transmissão é realizada ponto a ponto. Nesse modelo o processo deve enviar a mensagem para cada um dos destinatários. No modelo *muticast* cada mensagem é enviada uma só vez e somente para os integrantes do um determinado grupo. Por último, no modelo *broadcast* a mensagem é enviada para todos os processos, e depois são filtradas pelos processos interessados naquela mensagem.

2.3 Interação Humano-Computador

O estudo das relações entre pessoas e computadores é denominado Interação Humano-Computador (IHC). Seu campo de estudo é multidisciplinar e envolve a Ciência da Computação, Psicologia, Sociologia, Línguas, Artes, Design entre outras áreas de estudo. O foco de estudo da IHC não está nas pessoas e nem nos computadores, mas na interface entre os dois. Esta interface é comumente referenciada como interface do utilizador e é formada por artefatos de hardware e software que auxiliam as pessoas no processo de interação com o computador.

Neste trabalho, nossa preocupação é com um sub-tópico de estudo da IHC: acessibilidade. O conceito de acessibilidade porém, é oriundo da arquitetura e posteriormente foi incorporado à área de IHC. Para melhor esclarecer o conceito de acessibilidade conforme pretendemos tratá-lo no contexto deste trabalho, formalizamos o mesmo a seguir.

2.3.1 Acessibilidade

Este termo é usualmente empregado para designar a capacidade de ser oferecido um determinado serviço a pessoas portadoras de necessidades especiais. Na área de arquitetura, o termo é usado para designar o quanto uma construção é acessível aos portadores de necessidades especiais [Preiser e Ostroff 2001], como paraplégicos, cegos, e outros. Para tornar uma construção acessível é preciso oferecer recursos, tais como rampas, barras de apoio, sinalizações sonoras, sinalizações visuais e outros recursos, que garantam o livre acesso desses usuários especiais ao ambiente.

Na grande área de Computação, o termo é estudado pela área de IHC (Interação Humano-Computador) e geralmente é utilizado para designar a capacidade de um determinado *software* em oferecer ferramentas que permitam que usuários com necessidades especiais possam interagir com a máquina [Dix et al. 2004]. O tipo mais comum de ferramentas dessa classe são as de síntese e reconhecimento de voz, que auxiliam os usuários cegos a interagir com o sistema.

Atualmente, dentro do conjunto de usuários com necessidades especiais, não estão incluídos apenas os portadores de debilidades físicas, mas também aqueles com debilidades de software, de hardware e de rede. Este ampliação no escopo dos usuários com necessidades especiais vai ao encontro da definição de inclusão digital. Em outras situações, o conceito de acessibilidade se mistura com conceito de projeto universal. Para separar estes conceitos vamos defini-los a seguir.

2.3.2 Inclusão Digital

A inclusão digital visa fornecer o acesso aos recursos digitais da atualidade a um grupo cada vez maior de usuários. Deste modo, podemos dizer que a inclusão digital busca democratizar o uso dos recursos digitais, como os computadores e a *Internet*. O meio comum para alcançar a inclusão digital é a criação de projetos e ações que facilitam o acesso de pessoas de baixa renda às Tecnologias da Informação e Comunicação. Contudo, a inclusão digital também se volta para o desenvolvimento de tecnologias que forneçam

acessibilidade para usuários portadores de necessidades especiais. Deste modo, pode-se dizer que a inclusão digital se vale das metodologias da acessibilidade para alcançar seus objetivos.

2.3.3 Projeto Universal

Projeto Universal, do inglês *Universal Design*, ou ainda Projeto Total, é uma sub-área de estudo da área de Projeto que enfoca o desenvolvimento de produtos, serviços e ambientes a fim de que sejam usáveis pelo maior número de pessoas possível, independente de idade, habilidade ou situação. Está diretamente relacionado ao conceito de sociedade inclusiva e sua importância tem sido reconhecida por governos, empresários e indústrias.

No Projeto Universal é estudada uma série de questões que geralmente não são abordadas em um projeto comum, pois é necessário considerar todas as possibilidades de uso por usuários muito diferentes. Isso inclui questões sociais, históricas, antropológicas, econômicas, políticas, tecnológicas, e principalmente de ergonomia e usabilidade.

2.3.4 Síntese de voz

Síntese de voz é o processo de produção artificial de voz humana. Um sistema desenvolvido para este propósito é denominado sintetizador de voz. Um sintetizador de voz pode ser implementado em software ou hardware. Existem pelo menos três métodos para síntese de voz: Texto-para-Fala (*Text-to-Speach*) [Sproat 1997], analogia terminal [Schröder 2001] e a síntese articulatória [Branco et al. 1997].

Um sistema de Texto-para-Fala converte texto em linguagem normal para voz e consiste de duas fases principais. A primeira é a análise do texto, onde o texto de entrada é transcrito em fonemas ou outra representação lingüística. Na segunda fase ocorre a geração de formas de onda da voz, onde a saída acústica é produzida destes fonemas. Estas duas fases são geralmente denominadas de síntese de alto e baixo nível, respectivamente.

Na técnica de analogia terminal, um conjunto de regras é usado para determinar os parâmetros necessários para sintetizar uma determinada expressão usando-se um sintetizador de formantes. Os formantes podem ser definidos como picos de energia em uma região do espectro sonoro. Enquanto o espectro de cada nota de um instrumento pode variar consideravelmente com a altura, as regiões dos formantes permanecem estáveis, seja qual for a freqüência da nota. Portanto, os formantes funcionam como uma espécie de assinatura de uma determinada fonte sonora. As regras definidas nesta técnica são geralmente usadas em conjunto com uma especificação da cadeia de fonemas que representa a expressão desejada.

A técnica definida como síntese articulatória da fala é produzida através da modelagem do sistema humano de produção de fala. Esta técnica envolve os modelos das articulações ligadas a produção de fala (língua, lábios, palato, glote, etc) e também às cordas vocais. Através de regras, estes modelos são modificados de modo a representar os fonemas. As regras refletem as restrições dinamicamente impostas à articulação pelas suas massas, músculos associados, viscosidade, etc.

2.3.5 Reconhecimento de Voz

Reconhecimento de voz é uma técnica computacional que permite construir dispositivos que transcrevam a fala em texto automaticamente. Em alguns aplicações essa transcrição é apenas uma etapa intermediária no processo de compreensão da fala, possivelmente terminando com ações em resposta ao que foi dito. Uma vez que a importância está na descoberta do sentido das palavras e não apenas na detecção de palavras é comum definir essa área de estudo como sendo Reconhecimento de Fala ao invés de Reconhecimento de Voz.

Capítulo 3

Trabalhos Relacionados

Integrar várias pessoas em um mesmo ambiente virtual é uma preocupação desde o surgimento dos **MUD**. Atualmente, a dificuldade não é integrar vários usuários num mesmo ambiente, mas sim integrar um grupo heterogêneo de usuários. Por ser um grupo heterogêneo, significa que cada usuário pode possuir uma máquina com capacidade de hardware e software diferente das outras. Os ambientes possuem um requisito mínimo para sua execução. Se uma parcela dos usuários não possui esses requisitos, esta parcela estaria automaticamente excluída da imersão no ambiente. Contudo, para não excluir ninguém deste grupo, é possível construir versões diferentes do mesmo ambiente, onde cada versão tem um conjunto de requisitos distinto. Note que essas versões podem atender a uma ampla gama de usuários do ambiente. Mesmo assim, existe um outro problema: as diferentes versões de uma aplicação qualquer podem ter *interfaces* de visualização gráficas diferentes, inclusive com dimensões espaciais diferentes. Isto pode acarretar uma discrepância no modo que cada usuário enxergar o ambiente. Se o ambiente possui uma bola vermelha, os usuários conectados pela *interface* textual deveriam enxergar uma descrição textual sobre essa bola. Do mesmo modo, usuários conectados por uma versão com *interface* 2D ou 3D deveriam enxergar uma representação gráfica desta bola.

Neste capítulo abordamos alguns trabalhos encontrados na literatura que discutem soluções para os dois problemas apresentados acima. Esses problemas são os mesmos abordados nesta disertação, e a solução apresentada é a de Interpercepção, a ser discutida nos próximos capítulos. As soluções apontadas pelos trabalhos discutidos neste capítulo serão comparadas com a nossa abordagem.

3.1 *Cross-Plataform*

O *Cross-platform* [Han et al. 2005] é um motor para o desenvolvimento de jogos 3D acessíveis por diferentes plataformas. O desenvolvimento deste motor está ligado ao conceito de jogos multi-plataforma, que são jogos que possuem versões para diferentes plataformas de vídeo-game e até mesmo versão para PC. Deste modo, o objetivo do motor *Cross-plataform* é unir os jogadores de um jogo multiusuário e multiplataforma em um ambiente de jogo compartilhado, mesmo que cada jogador esteja conectado por uma plataforma de jogo diferente. O jogador também tem flexibilidade para trocar de plataforma a qualquer momento, sem perder o jogo que ele havia desenvolvido. Se o

jogador possui uma plataforma X e sempre acessa o jogo desta plataforma, mas ele viaja para um local onde só está disponível uma plataforma B, ele poderá continuar seu jogo normalmente da plataforma B. Quando um jogo é desenvolvido com este motor, ele adquire uma infra-estrutura que permite que o mesmo seja executado em diferentes plataformas (modelos) de video-games, além de ser capaz de se interligar a um ambiente multijogadores a partir de qualquer plataforma.

Além do motor *Cross-plataform* definir uma arquitetura de software para garantir a interligação dos jogos multi-plataforma, esta arquitetura usa um único servidor para tratar as informações enviadas por todos os usuários. Assim, ele garante o compartilhamento do ambiente de jogos entre todos os usuários. Esta arquitetura também define uma camada de **midleware** para comunicação entre o servidor e as diferentes plataformas de jogo.

O uso de um servidor unificado é semelhante à solução que abordada no presente trabalho. Contudo, o *Cross-plataform* esta focado no desenvolvimento apenas de jogos com *interface* 3D, ao contrário da nossa abordagem que trata da interligação de ambientes com versões em várias *interfaces* de visualização.

3.2 *RUNESCAPE*

Desenvolvida pela empresa britânica JAGEX, o *RUNESCAPE* é um jogo de aventura *on-line* massivo [West 2007]. Escrito na linguagem Java, o *RUNESCAPE* é executado como um *applet*, sendo acessado pelos seus usuários diretamente da *Internet*, através de um navegador, sem a necessidade de instalações adicionais. Com gráfico tridimensional sendo executado em tempo real e com características que lembram os atuais **MMORPG**, o *RUNESCAPE* tem atraído muitos jogadores (cerca de 6 milhões de jogadores registrados, segundo dados da própria empresa). Para poder se conectar ao ambiente virtual do *RUNESCAPE*, o usuário deve se cadastrar no *site* do jogo, criando uma conta de usuário, que pode ser gratuita ou paga. Após a abertura da conta o usuário deverá construir seu *avatar* através da composição de um conjunto básico de roupas, cor da pele e cabelos. Então ele estará pronto para iniciar o jogo. O jogo se passa em um mundo fictício habitado por monstros, magia e heróis típicos dos mais famosos jogos de **RPG**. O personagem é capaz de evoluir através do treinamento de suas habilidades, que vão desde "corte de lenha" até o "uso de magias". Os jogadores se comunicam através de *chat* textual, e podem realizar negociações de equipamentos virtuais, alianças e até mesmo travarem um combate, o que geralmente é visto como uma má conduta pelos moderadores do jogo. A figura 3.1 mostra um diálogo entre o *avatar* de um jogador e *avatar* de um **NPC** (*Non-Player Character*).

O *RUNESCAPE* possui dois modos de visualização. Ele pode ser executado com 64 Mb de RAM e um processador 300 Mhz e não necessita de uma conexão de banda-larga, sendo executado normalmente através de uma conexão discada. O usuário também pode optar por executar o jogo com uma melhor resolução (128 de RAM e processador de 500Mhz) o que aumenta o nível de detalhe do jogo. Independente da *interface* que seja escolhida, os usuários compartilham o mesmo ambiente.

Semelhante a nossa abordagem, o *RUNESCAPE* é executado diretamente pela *web*, utilizando um *applet*. Ele permite que o usuário escolha dois tipos de *interface* e garante

Figura 3.1: Interface do RUNESCAPE

que o mesmo compartilhará o ambiente de jogo com os demais participantes. Contudo, as duas versões disponíveis no *RUNESCAPE* são ambas com *interface* de visualização 3D, ao contrário da nossa, que prevê o uso de outros tipos de *interface* de visualização.

3.3 City

O *City* [Brown et al. 2003] é um sistema que permite a integração dos usuários reais e virtuais de um museu. Para a validação deste sistema, foram desenvolvidas duas versões virtuais da galeria *Mackintosh Interpretation Centre in The Lighthouse*, do Centro Escocês de Projeto, Arquitetura e Urbanismo. Uma das versões virtuais é um ambiente 3D e a outra é um pagina *web*.

Figura 3.2: Versão 3D do *City*

Os visitantes do ambiente 3D navegam através de uma *interface* de visualização com visão em primeira pessoa. Os avatares dos usuários *web* e dos usuários reais são representados simultaneamente nesse ambiente. Esses usuários se comunicam com os demais através de uma ferramenta de *Chat* acoplada à sua *interface* de visualização. A figura 3.2 mostra a *interface* de visualização 3D do *City*.

Os vistantes *web* interagem através de uma página HTML que exibe vários *links* com pontos de interesse dentro do museu. Esses usuários também contam com um mapa topológico bidimensional que exibe as posições dos usuários reais e dos usuários conectados através da *interface* de visualização 3D. Para se comunicar, esses usuários também utilizam uma ferramenta de *chat*. A figura 3.3 mostra a *interface* de visualização *web* do *City*

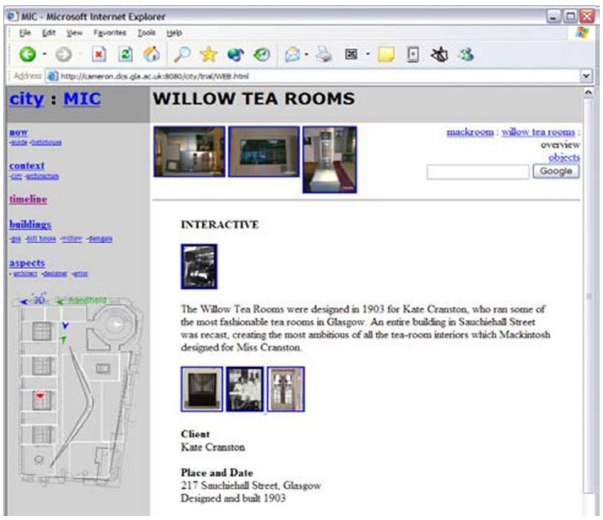

Figura 3.3: Versão *web* do *City*

Já os visitantes reais, utilizam um PDA e um *tracker* para interagir com os usuários virtuais. Através do PDA, os usuários reais podem acessar um aplicativo de *Chat* para comunicação. Nesse mesmo PDA, uma aplicação executa o mesmo mapa topológico exibido na *interface* de visualização *web*. Através do *tracker*, a movimentação dos usuários reais é capturada e transmitida para ser atualizada no ambiente 3D e no mapa topológico. A figura 3.4 mostra um usuário real do *City* e a *interface* de visualização apresentada através do PDA.

Apesar do foco do *City* ser o uso de realidade mista para integrar usuários reais e virtuais de um museu, ele emprega a proposta de permitir visualizar um ambiente a partir de *interfaces* de visualização distintas e integrar os diversos usuários deste ambiente em um mesmo ambiente compartilhado. Essa proposta aproxima o *City* do nosso trabalho, contudo, faz a diferenciação entre usuários de *interfaces* distintas.

Figura 3.4: Versão PDA do *City*

3.4 Nirve

O *Nirve* [Sebrechts et al. 1999] é uma ferramenta de busca que permite ao usuário realizar suas buscas através de três *interfaces* de visualização gráficas distintas: 3D, 2D e textual. Os usuários podem realizar as mesmas buscas e obter os mesmo resultados. A única diferença esta na apresentação destes resultados e nos recursos disponíveis para visualizá-los e gerencia-los. Na *interface* de visualização textual se pode fazer muito pouco, além de ver os dados. Já na *interface* de visualização 2D é possível marcá-los, movê-los, observar uma distribuição espacial e uma série de outras facilidades para auxiliar a visualização. A *interface* de visualização 3D apresenta os mesmos recursos da 2D, além de apresentar os dados com uma noção espacial de profundidade. A figura 3.5 mostra as três *interfaces* de visualização do Nirve. Da esquerda para direita estão as *interfaces* de visualização 3D, 2D e textual, respectivamente.

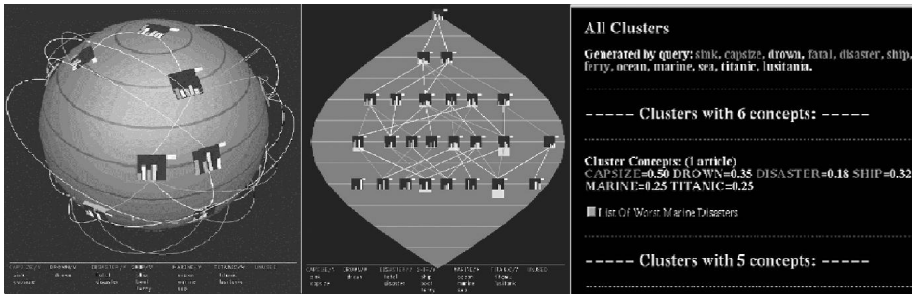

Figura 3.5: Intefaces do Nirve. Da esquerda para direita: *interface* de visualização 3D, 2D e textual.

Apesar de ser um aplicativo voltado para visualização de dados, o NIRVE ainda passa a idéia do uso de *interfaces* de visualização distintas para visualização do mesmo escopo de informações. Por isso, incluímos o mesmo como um exemplo rudimentar de criação

de um ambiente de interpercepção, mesmo que não haja comunicação entre os usuários. Dos trabalhos aqui apresentados, o NIRVE é o que apresenta melhor a idéia de uso de *interfaces* de visualização distintas para exibir o mesmo conteúdo.

3.5 Knowescape

Muitos usuários podem acessar a mesma página *web* simultaneamente. Algumas dessas páginas exibem quantos usuários estão acessando essa página naquele momento e se a página tiver serviços que exigem *login*, é possível exibir um apelido (*nickname*) dos usuários que executaram *login*. Contudo, estes usuários em geral não podem interagir uns com os outros. O *Knowscape* [Babski e Carion 2003] tem justamente o intuito de permitir a interação entre esses usuários.

Esse aplicativo converte uma página na *web* em um ambiente 3D multiusuário. Nesse ambiente 3D, os usuários são representados por seus endereços de IP e desse modo podem perceber os demais usuários que estão acessando aquela página. Os usuários podem criar espaços dentro do ambiente 3D onde eles podem colocar *links* de interesse criando comunidades virtuais para compartilhar o conhecimento armazenado nesses *links*.

O *Knowscape* é uma aplicação cliente-servidor baseado na *web* e desenvolvido na linguagem Java. Através de uma ferramenta de *Chat* acoplada à sua *interface* de visualização, o *Knowscape* permite que seus usuários se comuniquem. Para facilitar a visualização de algumas páginas o *Knowscape* mantém a versão original do página aberta, em uma aba de sua *interface* de visualização. Além disso, o *Knowscape* permite que a *interface* de visualização 3D seja acessada por aplicações clientes distintas e o conteúdo 3D seja criado em diferentes linguagens de modelagem, sendo a navegação final totalmente transparente ao usuário. A figura 3.6 mostra a *interface* de visualização do *Knowscape*.

O *Knowscape* também permite integração com usuários que acessem a *Internet* via celular. Quando estes usuários acessam uma página pelo *Knowscape* eles são representados no ambiente 3D por seu número de celular. Esses usuários podem se comunicar com os demais através de mensagens de SMS.

Mesmo que o foco central do *Knowscape* seja o de integrar usuários da *web* em ambiente virtual 3D, ele busca proporcionar essa integração por diversas *interfaces* de visualização, o que aproxima o *Knowscape* da Interpercepção. Além disso, o *Knowscape* utiliza a linguagem Java e a arquitetura cliente-servidor no seu desenvolvimento, assim como propomos nessa dissartação.

3.6 Comparativo Entre os Trabalhos

A Tabela 3.1 mostra um quadro comparativo entre os trabalhos discutidos. Os trabalhos foram agrupados nas colunas, sendo a última coluna destinada à nossa abordagem: a Interpercepção. Deste modo, buscamos traçar este comparativo entre as várias abordagens apresentadas e também já situar a nossa neste panorama. Definimos um conjunto de propriedades das aplicações: *Interface* de visualização (1D (textual), 2D e 3D), Comunicação, Acessibilidade, Ambiente Multi-usuário, Ambiente Massivo, Ambiente Compar-

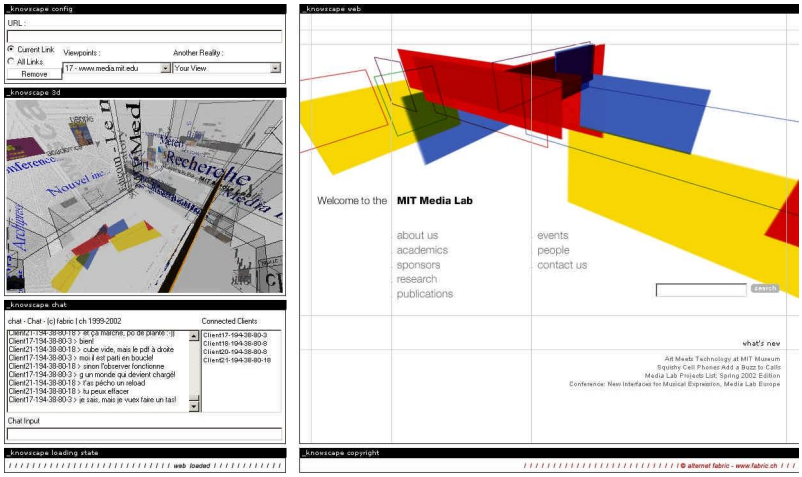

Figura 3.6: *Interface* de visualização do *Knowscape*

tilhado. A *interface* de visualização diz respeito aos tipos de *interface* para visualização das diversas abordagens. Como demonstra a Tabela 3.1, apenas a nossa abordagem possui os três tipos de *interface* apontados e trabalha no escopo dos ambiente multi-usuários. A Comunicação refere-se ao fato de uma abordagem possuir ou não uma ferramenta para comunicação. Todos os ambientes listados na Tabela 3.1 que são multiusuários, também possuem alguma ferramenta de comunicação acoplada às suas *interfaces*. A propriedade Ambiente Massivo refere-se à capacidade do ambiente em suportar uma quantidade massiva de usuários. A propriedade Ambiente Compartilhado refere-se à propriedade de interligar as várias *interfaces* de visualização do sistema para formar um único ambiente compartilhado. Acessibilidade refere-se ao fato da abordagem possuir alguma ferramenta de acessibilidade, tal como síntese e reconhecimento de voz.

Abordagem	**Runescape**	**City**	**Nirve**	**Knowescape**	**Cross-Plataform**	**Nossa**
Interface 1D	não	não	sim	não	não	sim
Interface 2D	não	sim	sim	sim	não	sim
Interface 3D	sim	sim	sim	sim	sim	sim
Comunicação	sim	sim	não	sim	sim	sim
Multi-Usuários	sim	sim	não	sim	sim	sim
Massivo	sim	não	não	não	sim	sim
Compartilhado	sim	sim	não	sim	sim	sim
Acessibilidade	não	não	não	não	não	sim

Tabela 3.1: Comparação entre os trabalhos relacionados

Capítulo 4

Interligação de Ambientes Virtuais

A *Internet* é um ambiente muito diversificado. Vários tipos de usuários navegam todos os dias por ela e buscam utilizar os variados recursos que estão disponíveis. Dentre estes recursos estão os ambientes virtuais.

Durante a evolução dos ambientes virtuais, as principais mudanças foram com respeito à *interface* utilizada para visualizar e interagir com o ambiente. Essas *interfaces* surgiram em versão textual, com o **MUD** e chegaram aos gráficos 3D no início da década de 90, com o *Active Worlds*. Mesmo que hoje existam ambientes virtuais com recursos gráficos 3D de alta definição, nem todos os usuários da *Internet* poderão interagir com esses ambientes, pois eles requerem recursos de hardware de última geração. Por isso, alguns desenvolvedores criam versões com diferentes tipos de *interface* de visualização, com intuito de atingir uma gama maior de usuários.

A construção de ambientes virtuais com diferentes *interfaces* para visualização garante que a estrutura do ambiente seja a mesmo em todas as versões. Neste caso, a única diferença entre cada versão é a *interface* utilizada para enxergar e conseqüentemente interagir com o ambiente. Este tipo solução prevê, portanto, que tanto a lógica de funcionamento do ambiente quanto as informações que não dizem respeito à visualização do ambiente, são imutáveis. Deste modo, podemos afirmar que esta solução segue o paradigma **MVC** (*Model-View-Controller*) para construção deste ambiente e suas várias versões. Discutiremos agora esse paradigma

4.1 Paradigma MVC

O paradigma **MVC** é uma arquitetura de software que separa os dados, *interface* de usuário, e controle lógico de uma aplicação em três componentes distintos de forma que as modificações para um componente possam ser feitas com o mínimo de impacto aos demais.

O **MVC** é freqüentemente referenciado como um padrão de *design* ou padrão de projeto de software. Contudo, o **MVC** trata mais profundamente da arquitetura de uma aplicação do que um padrão de *design* típico. Conseqüentemente o termo padrão arquitetônico pode ser útil [Buschmann et al. 1996]. Neste capítulo escolhemos tratá-lo como um paradigma, pois este foi o termo utilizado para defini-lo num dos primeiros artigos sobre o **MVC** [Burbec 1987].

O paradigma **MVC** foi descrito primeiramente em 1979 por Trygve Reenskaug, pesquisador da Xerox PARC, durante o desenvolvimento da *interface* do Smalltalk-80.

No paradigma **MVC**, as interações do usuário, a modelagem do mundo externo, e o *feedback* visual, para o usuário, estão explicitamente separadas. Elas são controladas por três componentes de *software* distintos. Cada um deste componentes de *software* é especializado na execução de sua tarefa. O *view* gerencia as saídas gráficas e textuais que são exibidas na *interface* visual da aplicação. O *controller* interpreta as informações transmitidas pelo usuário através dos periféricos de entrada e se encarrega de transmitir estas informações ao *view* ou ao *model*, caso seja necessário. Finalmente, o *model* administra o comportamento e os dados de domínio da aplicação, responde aos eventuais pedidos de informação sobre o seu estado atual (geralmente requisitados pelo *view*), e transmite instruções sobre eventuais mudanças de estado (geralmente requisitadas pelo *controller*). A figura 4.1 demonstra o funcionamento do **MVC**.

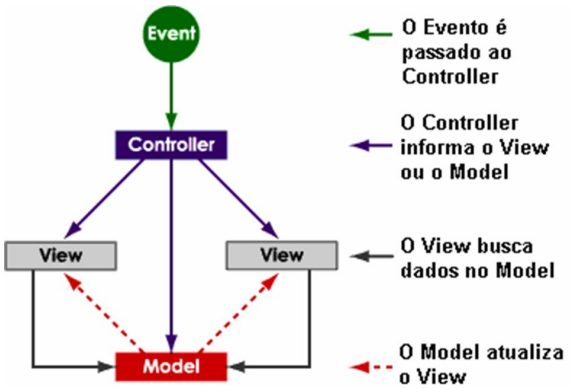

Figura 4.1: Funcionamento do **MVC**

Seguindo o paradigma **MVC** para construir uma versão com uma nova *interface* de visualização basta mudar o *view* e manter os demais módulos. Este é tipo de solução empregada pelo Nirve, apresentado na seção 3.4. No caso do Nirve, os dados que são visualizados são os mesmos, portanto, a mudança ocorre apenas na hora de exbir os dados.

Os dados trabalhados pelo **Nirve** são estáticos, o que demanda um estrutura de dados simples para o armazenamento e recuperação destes dados. Contudo, ambientes virtuais possuem dados geométricos que sofrem alterações dinâmicas em seu estado: mudam de forma, de posição, entre outras características. Para armazenar tais dados, será necessário alguma estrutura de dados geométricos apropriada. Ambientes com dimensões diferentes demandam estruturas de dados geométricos diferentes. Por isso, utilizar apenas o **MVC** não servirá para solucionar o nosso problema. Para melhor ilustrar essa afirmação vamos analisar algumas estruturas de dados geométricos.

4.2 Estruturas de Armazenamento de Dados Geométricos

Uma grande preocupação no desenvolvimento de ambientes virtuais está relacionada com a quantidade de memória utilizada para a execução do ambiente. O uso excessivo de memória pode comprometer o desempenho do ambiente, algo que é indesejável, principalmente em ambientes voltados ao entretenimento, como os jogos. Os desenvolvedores desses aplicativos consolidaram ao longo dos anos, estruturas de dados que otimizavam o uso da memória. Dessas estruturas, as mais difundidas foram os mapas de *tiles*, que são direcionados para aplicações 2D e os grafos de cena, que são voltados para aplicações 3D. Falaremos agora dessas estruturas em separado.

4.2.1 Mapa de Tiles

Um *mapa de tiles* é uma grande imagem composta por pequenas imagens menores chamadas de *tiles*. Para construir o mapa, o mesmo é dividido em blocos com as mesmas medidas. Essas medidas são as mesmas do *tile*. Para cada bloco um *tile* é mapeado. A figura 4.2 mostra um exemplo de *mapa de tiles*.

Os mapas são geralmente compostos com um número reduzido de *tiles*, que formam uma espécie de "paleta de *tiles*". Cada *tile* tem uma função dentro do mapa, por exemplo, representar os cantos, representar portas, telhados, entre outros objetos gráficos. O uso de uma "paleta de *tiles*"reduz a quantidade de imagens necessárias para compor um mapa. Além disso, as imagens são referenciadas para os blocos que vão ocupar no mapa, podendo uma mesma imagem ser referenciada para vários blocos, o que reduz o uso de memória.

Figura 4.2: Exemplo de Mapa de Tiles

A economia de memória alcançada pelo *mapa de tiles* favorece o uso dessas estruturas na construção de ambientes 2D, principalmente para jogos. Os computadores e vídeogames de gerações passadas possuíam pouca capacidade de memória. Era necessário

economizar a memória para garantir a execução dos jogos de maneira atraente aos usuários, uma vez que o estouro de memória causa problemas ao jogo, afastando os usuários.

4.2.2 Grafo de Cena

Um grafo de cena [Foley et al. 1995] é uma estrutura de dados composta de nós e arcos, onde os nós representam elementos de dados, e os arcos representam relações entre estes elementos de dados. Esses elementos de dados criam objetos que definem a geometria, som, luzes, local, orientação, e aparência dos elementos visuais e auditivos que irão compor o ambiente virtual. O grafo de cena é, portanto, uma junção de todos os elementos que compõem um ambiente. A figura 4.3 mostra um exemplo de grafo de cena.

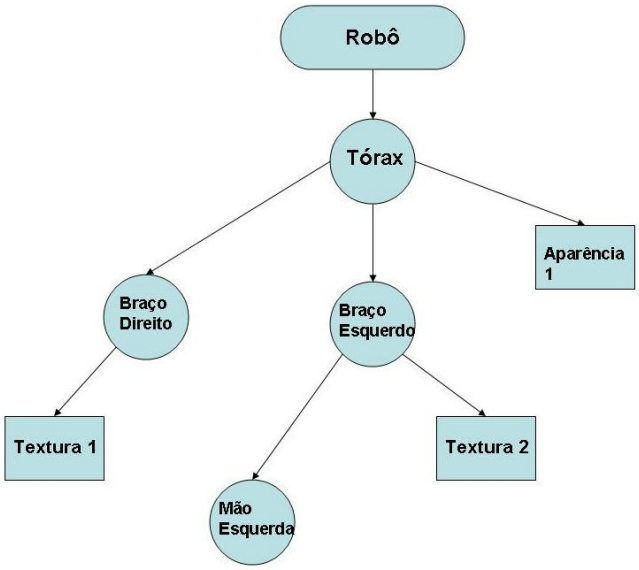

Figura 4.3: Exemplo de grafo de cena

No grafo de cena os nós são classificados quanto à hierarquia em nós pais e nós filhos. Os nós filhos herdam todas as características do seu nó pai. Por exemplo, se um nó pai possui uma determinada rotação aplicada a ele, todos os nós que são seus filhos herdaram essa mesma rotação. Essa propriedade do grafo de cena otimiza a construção de uma cena, pois permite que vários objetos que possuem alguma relação compartilhem uma determinada característica. Esse modelo de herança possui a restrição de que um nó qualquer só pode ter um pai, mas nada impede que um nó tenha vários filhos.

O grafo de cena usa técnicas importantes como eliminação de objetos que não contribuem para o resultado final da imagem (*culling*) e ordenação por estados de *hardware*. Essas técnicas conferem melhor desempenho ao sistema. A estrutura oferecida pelo grafo de cena faz com que seja possível aumentar a complexidade da simulação de maneira controlada, garantido escalabilidade ao ambiente. Por essas propriedades o grafo de cena é uma estrutura bastante utilizada em aplicações com gráficos tridimensionais.

4.2.3 Construção de Ambientes Semelhantes com Estruturas de Dados Distintas

Na seção 4.2.1, foi mostrado que para criar ambientes com interface de visualização 2D o *mapa de tiles* é mais viável. Na seção 4.2.2 mostramos que para ambientes com *interface* de visualização 3D o grafo de cena é a melhor escolha. Utilizar a mesma estrutura de dados para aplicações com *interfaces* de visualização com dimensões espaciais distintas pode comprometer o seu desempenho. O grafo de cena é uma estrutura complexa, e seu processamento pode tornar uma aplicação 2D mais lenta, uma vez que várias imagens podem ser utilizadas para desenhar um mapa. Um *mapa de tiles* é uma estrutura muito simples, mas que não seria capaz de resolver com eficiência a construção de uma cena com gráficos 3D.

Usar o **MVC** como padrão para codificação do ambiente pode não ser viável segundo os princípios do próprio padrão. Segundo a proposta desse padrão para criar uma versão com uma nova *interface* bastaria modificar o módulo *view* desse sistema. Mas como demonstramos, *interfaces* de visualização que utilizam gráficos de dimensões distintas fogem a essa regra, pois necessitam de estruturas de dados distintas para otimizar seu armazenamento. Para solucionar isso sem abandonar o **MVC**, alterações podem ser feitas nos outros módulos, para utilizar estruturas de dados distintas para cada interface. De certo modo, isto fere a proposta deste paradigma, o que nos leva a pensar se ainda seria necessário utilizá-lo. Se tratarmos o **MVC** como um padrão de projeto também, nos lembraremos que sua principal função é organizar o código da aplicação. Sendo assim, o **MVC** pode ser realmente a solução para criar várias versões do mesmo sistema, mesmo que tenhamos que modificar mais de um módulo da apliacação. Isto se deve ao fato de que utilizando este paradigma o código da aplicação permanecerá organizado, o que facilitará futuras alterações do mesmo.

Mesmo que tenhamos várias versões com *interfaces* de visualização distintas para uma dada aplicação, para torná-la uma aplicação multiusário, outras implicações surgem. Os usuários conectados em uma versão 2D, por exemplo, não podem interagir com os usuários conectados por outra versão. Mesmo que o ambiente modelado seja o mesmo em todas as versões, cada versão é percebida pelos usuários como uma instância distinta do mesmo ambiente. Para que haja interação entre os usuários de versões distintas é preciso intercambiar as mensagens entre as diferentes versões. Porém, as mensagens podem não ser compatíveis e por isso deverão ser convertidas.

4.3 Conversão de Mensagens

Quando um grupo de usuários está conectado por uma ferramenta de bate-papo, e um determinado usuário envia uma mensagem de texto aos demais, essa mensagem é enviada ao servidor e o mesmo encaminha a mensagem por *broadcast* aos demais usuários. Esse exemplo ilustra um aplicativo que segue a arquitetura cliente-servidor. Este modelo de funcionamento é o mais utilizado na construção de ambientes virtuais *on-line*. Num ambiente virtual *on-line*, cada usuário possui em sua aplicação cliente uma instância do ambiente. O servidor se encarrega de atualizar os eventos dinâmicos que ocorrem neste

ambiente, como o movimento de *avatares* e troca de mensagens de texto.

Independente da forma como os usuários enxergam o ambiente, as mensagens de texto trocadas entre os usuários seguem o mesmo formato, e são transmitidas da mesma forma. Desse modo, para unir em um mesmo ambiente, usuários conectados através de interfaces de visualização distintas, bastaria permitir que os mesmo compartilhassem a mesma ferramenta de comunicação textual. Esta mesma situação também é válida para comunicação por mensagens de áudio. Esse é o tipo de solução empregada pelo *City*, apresentado na seção 3.3. Acoplado a cada interface do *City* existe uma ferramenta de bate-papo através de áudio. Deste modo, em qualquer *interface* de visualização que o usúario esteja, ele será capaz de se comunicar com os demais. A ferramenta de comunicação neste caso, serve como uma ponte interligando as várias versões do ambiente.

Em muitos ambientes virtuais os usuários também interagem através dos seus *avatares*. Por meio deles os usuários percebem uns aos outros. Se o ambiente foi desenvolvido com gráficos 2D, então os *avatares* também serão 2D. Se existem usuários que estão conectados ao ambiente na versão 3D, os usuários do ambiente 2D não poderão enxergá-los, já que a sua interface não suporta gráficos 3D.

A percepção dos *avatares* é o obstáculo crucial para união de ambientes com *interfaces* de visualização diferentes. É necessário utilizar um processo de conversão entre os *avatares* das diferentes versões do ambiente. Esta conversão deve ser feita em duas ocasiões: na adição de *avatares* a um ambiente a na movimentação dos mesmos.

4.3.1 Conversão de *Avatares*

Quando um novo usuário entra no ambiente que utiliza arquitetura cliente-servidor, o servidor se encarrega de adicioná-lo na aplicação cliente dos demais usuários que estão conectados. Do mesmo modo, esse usuário que acabou de entrar receberá uma lista com os *avatares* que serão adicionados no seu cliente.

Avatares de qualquer *interface* de visualização possuem características básicas, tais como o nome do *avatar* e uma representação gráfica do mesmo. O nome dele independe da *interface* de visualização, mas a representação gráfica, não. Essa representação gráfica determina um modelo, que é definido pela própria arquitetura do ambiente. Este modelo é algo como um esqueleto do *avatar*, ou um *avatar* padrão (*default*). Em cada *avatar*, este modelo deve ser interpretado como um modelo geométrico que todos os *avatares* seguem. A individualização dependerá das adaptações possíveis, e será definidas pela preferência de cada usuário. Em geral, essa personalização ocorre por sobreposições de texturas no modelo geométrico do *avatar*. A figura 4.4 apresenta uma ferramenta de personalização de *avatares*, chamada Hero-O'matic [Reis 2007]. Nesta figura, a imagem da esquerda mostra o modelo geométrico do *avatar* e a imagem da direita mostra um *avatar* personalisado.

Se cada ambiente possui um *avatar* padrão é possível converter um *avatar* padrão de uma ambiente qualquer para o *avatar* padrão de outro. Se um determinado *avatar* possui personalizações, essas podem ser armazenadas antes de uma conversão. As personalizações são removidas, e o *avatar* se torna o padrão. Ele é convertido para o *avatar* padrão de outro ambiente. As personalizações são então acessadas. Caso seja possível aplicar

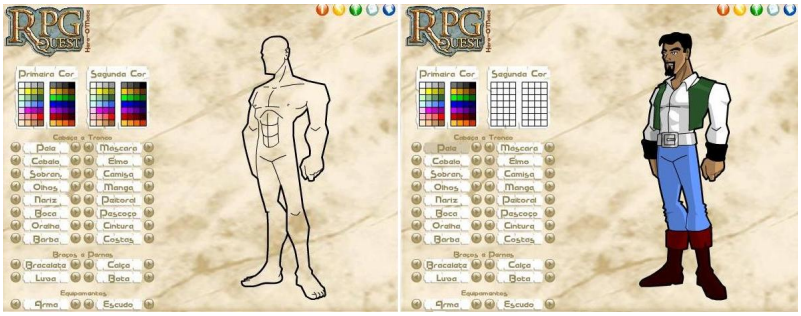

Figura 4.4: Ferramenta Hero-O'matic

aquele tipo de personalização elas são aplicadas, do contrário, elas podem ser convertidas ou descartadas. Se elas forem descartadas significa que o *avatar* não será idêntico, contudo esse não é o objetivo da conversão. Na verdade, o objetivo é transpor a representação de um *avatar* de uma *interface* de visualização para outra, para que ele possa ser percebido em outra *interface* de visualização. Por isso, o termo conversão pode até não ser o mais indicado neste tipo de operação, e sim o termo adequação. No Capítulo 2, os trabalhos *City* e Knowescape usam esse tipo de solução para converter *avatares* de uma *interface* de visualização para outra.

4.3.2 Conversão de Mensagens de Movimento

O movimento dos *avatares* em um ambiente virtual é calculado através de interpolação. Quando o *avatar* se desloca pontos de sua trajetória são capturados para o cálculo dessa interpolação. Esses pontos possuem coordenadas espaciais que são necessárias ao cálculo. Um ambiente 2D possui duas coordenadas, enquanto o 3D possui três.

As medidas de um mapa também podem variar. Mesmo que dois ambientes usem mapa de *tiles* como técnica de construção de ambientes, as medidas do bloco podem variar de um ambiente para outro. Isso pode impedir que um *tile* usado na construção de um ambiente seja aproveitado na construção de outro. Contudo, com o uso de escalas é possível definir equações de conversão entre os espaços. Estas conversões podem até ser feitas entre ambientes com gráficos de dimensões diferentes. Ao converter do 2D para o 3D uma coordenada deve ser adicionada, no caso com um valor padrão e no caminho inverso uma coordenada deveria ser descartada.

Um dos problemas cruciais da conversão entre ambientes com dimensões distintas é a conversão de um ambiente 1D (textual) para os demais. A dificuldade está em converter um dado impreciso como um texto em coordenadas espaciais e imagens bi e tridimensionais. Estas coordenadas são vitais para o deslocamento dos *avatares* nas *interfaces* de visualização 2D e 3D. Um meio de contornar este problema é formalizar coordenadas espaciais na interface 1D. Formalizar todos os pontos existentes nas demais *interfaces* pode ser um trabalho desnecessário uma vez que o a noção de espaço é abstrata na *interface* textual. Contudo a definição de alguns pontos é importante para geração do movimento nas outras *interfaces*.

Outro ponto problemático na *interface* de visualização 1D é a navegação pelo ambi-

ente. Assim como tudo neste *interface* ela deve ser trabalhada na forma de texto e forma mais viável e definição de uma linguagem formal para interação com o ambiente. Este tipo de estratégia já era usada nos primeiros **MUD**'s e continua sendo usada até hoje neste tipo de aplicação. Esta linguagem formal define todos os comandos disponíveis para uso no ambiente. Sendo uma linguagem formal, ela deve possuir uma grámatica formal que define de forma precisa as regras para uso dos comandos deste *interface* de visualização. Esta linguagem também é imporantente para o eventual uso de ferramentas de comunicação tal qual as ferramentas de síntese e reconhecimento de voz.

Capítulo 5

O Sistema de Interpercepção

No capítulo anterior, apresentamos os vários problemas e soluções acerca da interligação de ambientes virtuais. Neste capítulo, apresentamos a nossa proposta: a interpercepção. O projeto da interpercepção partiu da existência das três versões do *PercepCom*. Para facilitar o nosso trabalho com o *Percepcom*, nós construímos uma arquitetura geral do sistema *Percepcom*, que é seguida na implementação de cada uma das versões. Esta arquitetura é discutida a seguir.

5.1 Arquitetura Geral do Percepcom

A arquitetura geral do *PercepCom* é mostrada na figura 5.1. Esta arquitetura segue o paradigma **MVC** e o componente Armazenamento de Dados, Visualização Gráfica e Controle, representam respectivamente os módulos *Model*, *View* e *Controller* do **MVC**. O módulo de Integração de Periféricos representa uma parte do controle que foi transformada em um módulo menor para diminuir a complexidade desta parte do sistema. Do mesmo modo, o módulo de Construção de Ambientes é uma parte especializada do módulo de Visualização, dedicada à tarefa de construção do ambiente virtual. O módulo de Integração tem a função de encapsular o sistema e permitir ao usuário abstrair o funcionamento de baixo nível do sistema. A presença do módulo Integração é uma analogia ao padrão de projeto Facede (Faixada) [Gamma et al. 1999].

A partir da versão original do *PercepCom*, que foi criada com *interface* de visualização 3D, foram criadas outras duas três versões: uma com *interface* de visualização 2D e outra com *interface* de visualização textual. Estas outras versões foram criadas pensando nos usuários que não possuiam recursos de hardware e software para executar a versão 3D do *PercepCom*. Nosso objetivo era conceder um maior grau de acessibilidade ao sistema.

As três versões do *PercepCom* trabalhavam em separado. Os usuários da versão 2D não eram capazes de se comunicar com os usúarios da versão 3D. Poderiamos interligar estas duas versões através de uma ferramenta de bate-papo, como propõe o trabalho City apresentado na seção 3.3. Contudo, queríamos que os usúarios compartilhassem também a informação visual gerada pelos seus avatares. Além disso, não queremos que algum usuário, de qualquer versão do *PercepCom*, sinta alguma forma de exclusão, por estar usando uma *interface* de visualização com recursos gráficos mais simples. Nossa meta é criar um ambiente que possa ser compartilhado por todos, independente de qual *interface*

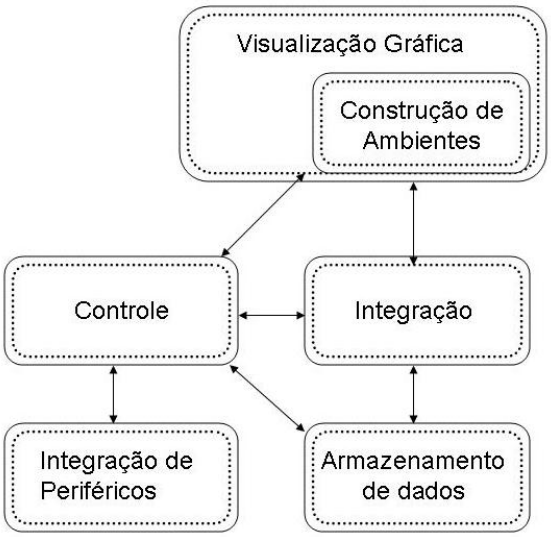

Figura 5.1: Arquitetura Percepcom

de visualização ele escolheu para acessar o ambiente. Queremos, portanto, interligar os ambientes das várias versões do *PercepCom* para construir um único ambiente compartilhado.

Para interligar o ambiente nós propomos uma teoria que denominamos de *interdimensionalidade*. Ela trata da união de ambientes com dimensões diferentes. A proposta da *interdimensionalidade* é converter as mensagens que são trocadas entre os clientes e o servidor do sistema. Para o melhor entendimento deste processo de conversão é necessário apresentar essas mensagens que são trocadas entre os clientes e o servidor.

5.1.1 Protocolo de Comunicação do Sistema PercepCom

Nesta seção são apresentadas as mensagens utilizadas nesse sistema, assim como um conjunto de tabelas que relatam a comparação entre as mensagens trocadas entre os três tipos de clientes (3D, 2D e 1D) e servidor unificado do sistema.

Na tabela 5.1, temos as mensagens de controle do sistema, que são enviadas pelo servidor para gerenciar o funcionamento do sistema. A **MCONNECT** é enviada pelo cliente para o servidor assim que ele entra no sistema pela primeira vez. Nesta mensagem é informado o *nick* (apelido) e o avatar do usuário. A mensagem **MNICKEXISTENTE** é enviada do servidor para o cliente no caso de seu *nick* já está em uso. A mensagem **MMUDARSALA** enviada pelo cliente pedindo ao servidor que o mude de sala dentro do ambiente. Estas mensagens de controle são idênticas nas três versões do sistema e não precisam passar por nenhuma conversão.

As mensagens de controle restantes, **MMUNDO** e a **MADDUSUARIO**, requerem alguma conversão pois seu formato difere entre as versões do sistema. A **MMUNDO** é a responsável pelo armazenamento de uma cópia do mundo virtual na aplicação cliente. Essa mensagem é enviada toda vez que ele entra em um ambiente diferente. Tal mensagem deve ser diferente para usuários conectados via *interface* de visualização 3D, 2D e

1D. Porém, essa diferença não está no formato da mensagem, mas sim no seu conteúdo, pois ela aponta para uma **URL** onde existe um arquivo em **XML** contendo os diferentes recursos 3D, 2D e 1D.

A mensagem **MADDUSUARIO** é usada para a adição de novos usuários depois que ambiente já esta em execução, ou seja, depois que uma mensagem **MMUNDO** já foi utilizada para trazer uma cópia do ambiente. É importante salientar que essa cópia do ambiente também trás uma lista dos usuários que estavam conectados no ambiente naquele instante. No caso do uso da mensagem **MADDUSUARIO** um novo usuário seria adicionado à lista.

Mensagens	**3D**	**2D**	**1D**
MCONNECT	nome: String type: char	nome: String type: char	nome: String type: char
MNICKEXISTENTE	Mensagem: String	Mensagem: String	Mensagem: String
MDISCONNECT	id: Integer	id: Integer	id: Integer
MMUNDO	mapName: String usuarios: Hashtable <Avatar3D> idMundo: int	mapName: String usuarios: Hashtable <Avatar2D> idMundo: int	mapName: String usuarios: Hashtable <Avatar1D> idMundo: int
MADDUSUARIO	nome: String avatar: Avatar3D id: Integer type: char	nome: String avatar: Avatar2D id: Integer type: char	nome: String avatar: Avatar1D id: Integer type: char
MMUDARSALA	idSala: int nome: String id: Integer	idSala: int nome: String id: Integer	idSala: int nome: String id: Integer

Tabela 5.1: Mensagens de Controle do sistema Interperceptivo

Além das mensagens de controle existem também as mensagens de movimento, as mensagens de texto e as mensagens de áudio. Estas três mensagens foram denominadas mensagens de interação e estão agrupadas na tabela 5.2. A mensagem **MTRANSFORM** delimita as mensagens de movimento. São usadas para realizar as transformações afins necessárias para representação do movimento dos avatares. Já **MTEXTO** é destinada ao envio de mensagens textuais para a comunicação ente os usuários. Por fim, a **MAUDIO** é usada para comunicação via áudio.

Mensagens	**3D**	**2D**	**1D**
MTRANSFORM	array[58]: byte	array[18]: byte	array[18]: byte
MTEXTO	id: Integer texto: String	id: Integer texto: String	id: Integer texto: String
MAUDIO	array[1000]: byte	array[1000]: byte	array[1000]: byte

Tabela 5.2: Mensagens de Interação do sistema IterP

Do grupo das mensagens de interação, apenas as mensagens de movimento requerem

alguma transformação. As mensagens de movimento no ambiente 3D são formadas por pontos com 3 coordenadas, que representam altura, largura e profundidade. O ambiente 2D do *PercepCom* é desenhado com visão de câmera isométrica e possui largura e profundidade. Já o ambiente 1D, é uma abstração descritiva dos demais. Para garantir a integração dos três ambientes a tarefa mais importante a ser realizada é a conversão do movimento realizado em um ambiente para os demais. Além disso, é necessário garantir que cada *interface* de visualização tivesse um modelo de avatar condizente com a sua dimenssão. Assim, surgiu a idéia de converter os avatares e os deslocamentos realizados por eles entre as diferentes versões do ambiente. Para aplicar esta idéia, desenvolvemos uma arquitetura de software, chamada arquitetura interD.

5.2 Arquitetura Interdimensional - InterD

A arquitetura *interdimensional* baseia-se no uso de um servidor unificado a ser usado por todos os tipos de clientes conectados ao sistema. Cada tipo de cliente representa uma das versões do *PercepCom*. O servidor aqui citado foi construído com base no *framework* H-N2N [Burlamaqui et al. 2006], que nos provê uma arquitetura hierárquica de servidores, o que possibilita a criação de ambientes virtuais massivos. A partir de modificações nesse *framework*, implementamos a arquitetura *InterD*. Esta arquitetura é apresentada na figura 5.2.

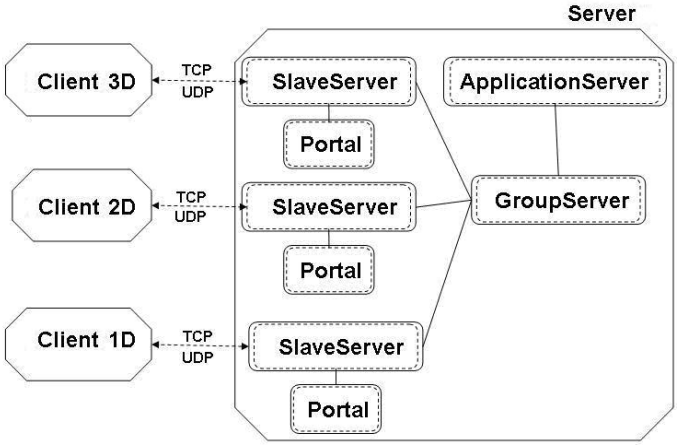

Figura 5.2: Arquitetura InterD

Nesta figura, vemos os Cliente 3D, 2D e 1D que correspondem aos aplicativos clientes das três *interfaces* de visualização do *PercepCom*. Os clientes representados nesta arquitetura são a junção de todos os clientes com aquele tipo de *interface* de visualização. Deste modo, o Cliente 2D nesta figura representa todos os usuários que utilizam uma aplicação cliente com *interface* de visualização 2D. Esta arquitetura também demonstra que aplicações clientes diferentes acessam o mesmo servidor e, portanto, trocam as mesmas informações. Se uma mensagem de texto é enviada por algum cliente 3D, ela poderá ser repassada para os demais tipos de clientes.

Os componentes *SlaveServer*, *ApplicationServer* e *GroupServer* pertencem ao *framework* H-N2N. Apresentaremos agora uma breve descrição dos componentes do H-N2N que são vitais para o entendimento do trabalho aqui discutido.

5.2.1 Componentes do H-N2N

O *ApplicationServer* é um componente responsável pela espera das conexões dos clientes. Ele atua como um servidor principal. O *SlaveServer* mantém uma comunicação com o cliente. Existe um componente deste tipo para cada cliente. Ele auxilia o *GroupServer* na tarefa de envio e recebimento de mensagens dos clientes. Por último, o *GroupServer* é um componente que contém a lista dos *SlaveServer's* que representam os clientes conectados. Ele é o responsável por gerenciar as trocas de mensagens entre os clientes de um determinado grupo. Ele também responde pelo processamento das operações de filtragem e junção. Além disso, ele pode se comunicar diretamente com outros *GroupServers* de mesmo nível.

Além dos componentes do H-N2N, existe no lado servidor da arquitetura InterD um quarto componente chamado PORTAL. Este componente não pertence originalmente ao H-N2N e trata-se de uma modificação necessária para a concepção deste trabalho. O PORTAL é responsável pelo processo de conversão das mensagens que são trocadas entre os clientes e o servidor.

5.2.2 O Componente *PORTAL*

Este componente é o responsável por transformar as mensagens que chegam ao *SlaveServer* de uma dimensão para outra, retornando-as ao cliente, como pode ser visto na figura 5.1.

Para ilustrar o funcionamento do *PORTAL*, suponha que um usuário conectado através da *interface* 3D, desloca-se no ambiente. Esse usuário geraria um fluxo de mensagens de movimento contendo informações como identificador do usuário e sua posição nas coordenadas x, y e z. Assim que tais mensagens chegarem ao servidor elas serão repassadas para todos os *SlaveServer* que então se encarregam de enviar as mesmas para os seus clientes. Porém, com a adição do componente *PORTAL* à arquitetura do H-N2N, algumas mensagens passaram por uma função de conversão para se adequarem ao seu cliente de destino. Esta função de conversão está implementada no *PORTAL*. Quando este componente recebe uma mensagem, ele verifica quais são as *interfaces* de origem e de destino, e, caso sejam diferentes, ele realiza a conversão adequada antes de repassar a mensagem.

O componente *PORTAL* foi implementado seguindo o diagrama de classe descrito na figura 5.3, construído com o padrão de projeto *Strategy* [Gamma et al. 1999]. Como evidencia a figura, existe um PORTAL para cada tipo de *interface* de visualização e todos eles são especializações de uma classe ancestral chamada PORTAL. Esta classe ancestral contém apenas os mecanismos para desempacotar as mensagens que ele recebe e verificar qual a interface de origem. Nas versões especializadas do PORTAL são definidas as regras para converter as mensagens.

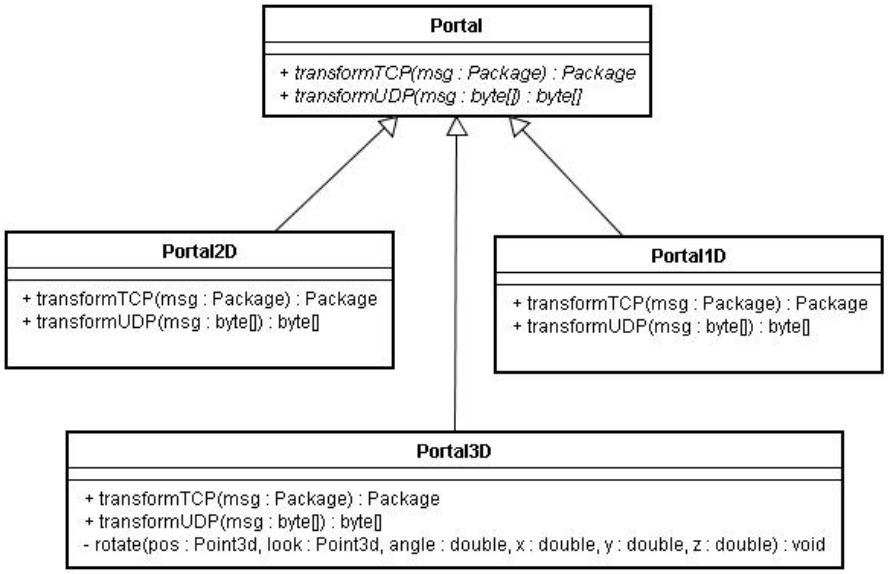

Figura 5.3: Diagrama de classes do Portal

Existe dois conjuntos de regras para conversão de mensagens: as regras gerais e as específicas. As regras gerais independem da *interface* de visualização e são:

- Cada PORTAL está acoplado à *intreface* do seu tipo, ou seja, o PORTAL 1D está acoplado à *interface* 1D;
- Cada PORTAL é capaz de converter uma mensagem originada de uma *interface* diferente para a *interface* do seu tipo, ou seja, o PORTAL 1D é capaz de converter mensagens da *interface* 2D e 3D para a *interface* 1D;
- Cada PORTAL deve ignorar a conversão, caso a *interface* de origem seja do seu tipo;
- Cada PORTAL é capaz de converter dois tipos de mensagem: as de movimento e mensagens relacionadas com a adição de avatares.

Como foi discutido na seção 5.1.1 as mensagens a serem convertidas são as que tratam do movimento e adição de avatares. As regras específicas do PORTAL tratam justamente dessas conversões e elas variam de acordo com a *interface*. Discutiremos em separado as regras de conversão das mensagens de movimento e das mensagens de adição dos avatares.

5.2.3 Conversão das Mensagens de Movimento

As mensagens de movimento são compostas por um vetor de pontos, que são usados para gerar um movimento através de interpolação. Para realizar a conversão do movimento realizado pelos avatares, trabalhamos inicialmente com a idéia de converter os pontos entre os espaços 2D e 3D. Para convertermos este espaços, utilizamos as seguintes regras:

- No espaço 2D, cada tile do mapa tem 64 pixeis;
- A posição do avatar no espaço 2D se dá em relação a parte superior esquerda da imagem;
- No ambiente 3D a posição é relativa ao centro do avatar;
- Os ambientes devem ter a sua origem na mesma posição;
- A área dos ambientes devem ser proporcionais para todos os objetos.

Como o espaço 2D começa das coordenadas (0,0), o ambiente 3D tem que ser modificado de modo que a coordenada mínima do plano onde o avatar se move (plano XZ na maioria dos casos) seja (0,0). Deste modo, a conversão ocorre apenas por projeção do plano onde os avatares se movem para o plano 2D. Existe um valor que é somado às coordenadas do ambiente 3D para que de o valor exato no 2D. Isso é necessário pois, a coordenada no sistema do ambiente 2D é a coordenada superior esquerda da figura do avatar. Já no sistema 3D a coordenada é o ponto onde o avatar está no plano, assim é necessário somar à coordenada do avatar 3D o tamanho da imagem em pixels para que a conversão se dê de forma direta. É necessário também que a cena 3D seja proporcional à cena 2D um número inteiro de vezes, para que a conversão não gere perdas.

Com essas informações, foi criado um conjunto de equações para conversão entre os espaços 2D e 3D. Uma posição no espaço 3D é definida por uma tupla <$3d.x$, $3d.y$, $3d.z$> e uma posição no espaço 2D é definido pela tupla <$2d.x$, $2d.y$>. Essas equações são:

$$\begin{aligned} pos3d.x &= \frac{pos2d.x + \frac{L_T}{2}}{R} \\ pos3d.y &= altura\ pre-definida \\ pos3d.z &= \frac{pos2d.y + \frac{L_T}{2}}{R} \end{aligned} \tag{5.1}$$

onde L_T é o tamanho do *Lado do Tile*, R é a razão entre as áreas dos ambientes 2D e 3D e altura pré-definida depende do tamanho do avatar;

Seguindo estas equações, se o avatar no espaço 2D estiver na posição <64,64>, no espaço 3D ele estará na posição <1.92, $altura\ pré-definida$, 1.92>. Portanto, mapeamos a posição X do espaço 2D para o X do espaço 3D, e mapeamos a posição Y do espaço 2D para o Z do espaço 3D. A posição Y do espaço 3D está pré-definida pois esta coordenada não é utilizada no espaço 2D. A metade de L_T define um ponto no centro da imagem do avatar 2D. Esta medida é necessária para calcular o centro da avatar 3D. A razão R define uma relação entre a quantidade de pixels no espaço 2D e a proporção equivalente em metros no 3D. A razão usada em nossos ambientes é a de 1 *pixel* para cada 2 cm. Deste modo, 64 pixels no espaço 2D equivalem a 1,28 metros no espaço 3D.

As equações citadas acima são para conversão dos espaços entre o ambiente 2D e 3D. Nestes ambientes o movimento dos avatares é essencial. Na interface 1D (textual) o movimento é apenas uma abstração. Os avatares no 1D são apenas um texto, assim como tudo neste ambiente. Quando um usúario do ambiente 1D deseja se desclocar pelo ambiente, ele deve utilizar usar uma palavra reservada, que delimita a movimentação. No *PercepCom 1D* esta palavra resservada quando usada deve ser seguida por um parâmetro, onde este parâmetro define o nome de um local para onde se deseja ir. A *interface* de

visualização *PercepCom* 1D informa que o usuário está se deslocando e ao chegar no local especificado, exibe a descrição associada a este local.

Para representar a movimentação dos avatares do 1D nas demais *interfaces*, foram definidos pontos de interesse no ambiente 1D. Estes pontos de interesse são os locais mais importantes de uma sala do ambiente. São nestes pontos de interesse que ocorrem a maioria das interações entre os usuários. Estes pontos de interesse foram delimitados com base na *interface* de visualização 2D, pois este *interface* é a que permite a visualização mais ampla do ambiente. A figura 5.4 ilustra a delimitação de dois pontos de interesse numa sala do ambiente 2D. Um primeiro ponto esta marcado numa região onde vários usuários estão interagindo no momento. O segundo ponto esta numa região mais central da sala.

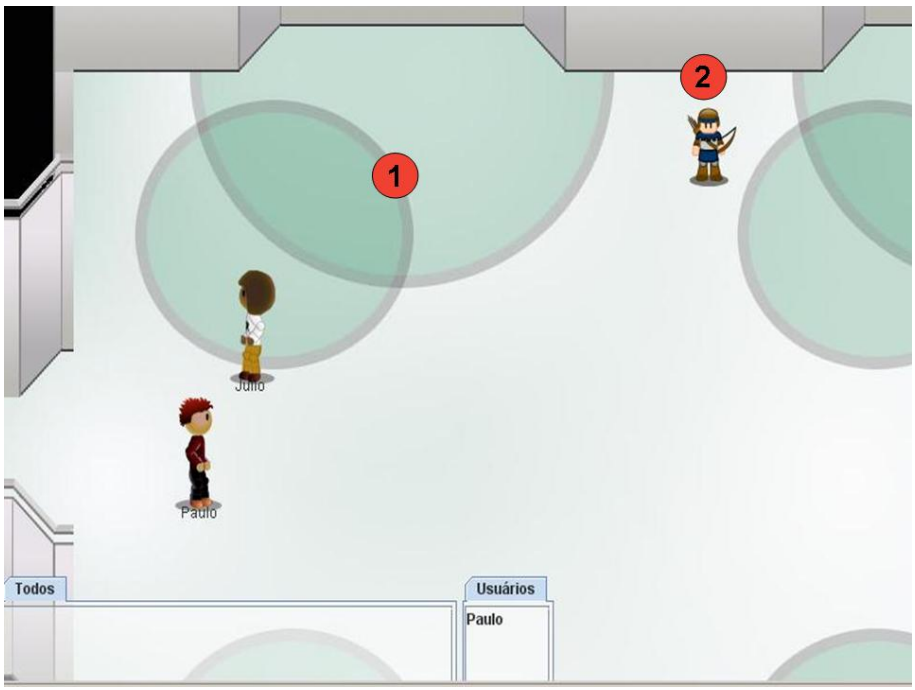

Figura 5.4: Pontos de interesse no 2D

Os dois pontos de intersesses delimitados na figura 5.4 são mapeados em um local do ambiente 1D e recebem um nome e uma descrição. A figura 5.5 mostra o resultado deste mapeamento. Nela pode ser visto o ponto 1 sendo referenciado com o nome *Circulos*. Já o ponto dois foi referenciado como *Fundo_Da_Sala*. Uma descrição será associada a cada um destes pontos. Quando algum usuário se aproximar ou examinar esse local, a *interface* de visualização do 1D irá exibir esta descrição.

No ambiente 2D os pontos de interesse possuem coordenadas espaciais. O ponto de interesse mapeado no 1D também receberá essas coordenadas. Deste modo, o ponto de intreresse pode ser definido como uma tupla <*nome*, *descrio*, *coordenadas2D*>. As coordenadas são necessárias para criar a movimentação do avatar 1D nos demais ambientes. Quando um usuário 1D decide ir até algum local de uma sala, ele verifica quais os pontos de interesse diponíveis naquela sala e escolhe um destes pontos para ser usado como

Figura 5.5: Pontos de interesse mapeados no 1D

parâmetro para a palavra reservada de movimentação no 1D. Os únicos locais para onde um usuário 1D pode se deslocar são os definidos pelos pontos de interesse. Assim, quando um usuário 1D realiza um movimento, ele está indo de um ponto de interesse para outro. Em posse do ponto de origem e do ponto de destino é possível criar uma interpolação, que simulará esta movimentação no ambiente 2D. A partir da movimentação no ambiente 2D é possível simular a movimentação no ambiente 3D seguindo o sistema de equações de conversão.

5.2.4 Conversão das Mensagens de Adição de Avatares

As mensagens para adição de avatares são compostas de uma tupla <*apelido*, *tipo*, *corpo*, *posição*, *orientação*, *destino*>. O apelido é o nome dado pelo usuário ao seu avatar. O tipo classifica os avatares em grupos, e é importante para delimitar o avatar no momento da conversão. O corpo define a estrutura física de um avatar. A posição representa o ponto espacial onde o avatar estará no momento em que for adicionado ao ambiente. A orientação delimita o ângulo de orientação para onde o avatar se desloca. O destino determina a *interface* de destino de uma mensagem.

O tipo de um avatar é uma característica que independe da *interface* de visualização. O tipo é usado como uma referência para determinar o corpo de um avatar no momento da conversão. O corpo por sua vez é dependente da *interface*. O corpo num ambiente 2D é representado por um vetor de imagens, geralmente definido na literatura como **Sprite** [Jones 2000]. O corpo no ambiente 3D é representado por um modelo 3D. No ambiente 1D o corpo é representado por uma descrição textual. A característica posição também

depende da *interface* de visualização. Essa característica já foi discutida na seção 5.2.3. A característica orientação é imprescindível no ambiente 3D, mas é desnecessária as demais.

O destino é uma informação usada apenas pelo servidor para determinar para qual PORTAL a mensagem deve ser enviada. Quando esta mensagem já está em algum PORTAL, este PORTAL determina a *interface* de origem a partir do dado contido no campo posição. Deste ponto em diante o PORTAL segue o algoritmo abaixo para conversão dos espaços.

Se *interface* de origem = *interface* acoplada ao PORTAL **então**{

Não realize a convresão;

}

Senão{

Encontrar o tipo do avatar

Descartar o Corpo do avatar

Com base no tipo criar um novo corpo adequadodo a *interface*

Converter a posição seguindo a equação de conversão

Se a *Interface* for 3D **então**{

Utilizar a orientação }

Senão então{

Descartar a orientação}

Como demonstra o algoritmo, as informações contidas na mensagem de adição de avatares são utilizadas para criar um avatar antes de sua adição. Cada *interface* de visualização tem um tipo específico de avatar. Todos estes avatares descendem de um avatar padrão. Na seção 4.3.1 discutimos as estratégias para conversão de avatares. Seguindo aquelas estratégias, definimos um avatar padrão chamada apenas de Avatar e depois criamos as especializações dele: Avatar 1D, Avatar 2D e Avatar 3D. Quando o PORTAL recebe uma mensagem para adição de avatares ele segue o algoritmo descrito acima. Através deste algoritmo o PORTAL cria um novo avatar, adequado a *interface* de visualização a qual ele esta acoplado.

5.3 A Interpercepção

O conceito de Interpercepção surge da idéia de que os usuários conectados ao nosso sistema percebem o mesmo ambiente e os mesmos usuários que estão no ambiente de formas diferentes. Essas formas diferentes são oriundas das várias *interfaces* de visualização que podem ser utilizadas para acessar o ambiente. Um dos principais objetivos da interpercepção é garantir que um usuário ao interagir com os demais não seja capaz de perceber ou distinguir a partir de qual *interface* os outros estão conectados. Isto é o que chamamos de interagir de forma transparente. Para garantir esse tipo de funcionalidade tivemos que aprimorar o conceito de *interdimensionalidade* para o conceito de interpercepção. Esse aprimoramento é na verdade um definição de novos conceitos que são agregados ao de interdimensão. Esses conceitos que definimos como as leis da interpercepção e são discutidos a seguir.

5.3.1 As leis da Interpercepção

Para agregar a Interpercepção a algum sistema, as seguintes leis devem ser adotadas:

- **Correspondência de Ambientes**: Cada ambiente é modelado tridimensionalmente, possui uma versão correspondente à sua planta baixa ou uma projeção isométrica, e uma descrição textual de suas características. Isso permite que em cada versão o mesmo ambiente seja visualizado, porém de uma forma adequada à sua interface;
- **Abstração de Mensagens**: Os usuários ao se conectarem a qualquer uma das interfaces conseguirão perceber mutuamente os demais de uma forma síncrona. Isso ocorre graças ao gerenciamento da navegação ser centralizado em um único servidor. Este servidor trata em um nível de abstração mais alto os dados referentes à movimentação e ao fluxo de usuários;
- **Adequação de Interface**: Nenhuma *interface* de visualização é capaz de suportar dados de outra *interface*. Por isso, para o usuário final, as informações exibidas devem ser adequadas ao tipo de *interface* de visualização a qual ele está acessando.

A lei da correspondência garante que um usuário, ao entrar no sistema, perceba o mesmo ambiente, independente de qual interface ele utilizou para o acesso. Desta forma, garantimos que este ambiente virtual seja compartilhado por todos os usuários, sem restrições. Cada usuário tem, portanto, a liberdade de escolher uma *interface* de visualização que seja adequada às suas necessidades. A fim de garantir a correspondência, as informações sobre o ambiente, em cada uma das suas versões (3D,2D e 1D), são armazenadas em um arquivo de matadados **XML**. Quando o servidor busca informações sobre o ambiente, ele vai até este arquivo previamente armazenado em um banco e a partir dele acessa os dados em si. Nesse arquivo os dados estão separados em informações diferentes, sobre a versão 1D, 2D e 3D. Quando a busca é realizada, o servidor deve definir qual *interface* de visualização está buscando, assim será retornado apenas os dados desejados para composição do ambiente na aplicação do cliente, e na versão suportada pela aplicação do cliente.

A lei da abstração diz respeito ao uso de um servidor unificado que é capaz de tratar as mensagens de todos os tipos de clientes, onde cada tipo de cliente corresponde a um tipo de *interface*. Todos estes cliente enviam suas requisições para um mesmo servidor. Este servidor é capaz de identificar a *interface* de visualização de cada cliente e proceder de forma adequada para resolução de cada requisição. O servidor unificado, portanto, é responsável por garantir que todos os clientes estarão interligados num mesmo ambiente, mesmo que os clientes sejam todos diferentes.

A lei da adequação diz respeito ao funcionamento do PORTAL, que é a ferramenta que garante a conversão das mensagens, a fim de garantir que elas sejam visualizadas de forma adequada. Como já foi dito anteriormente, o PORTAL é um componente do servidor. A lei da adequação está intimamente ligada à lei da abstração, já que ambas são resolvidas pelo servidor. Contudo, a lei da abstração serve apenas para garantir que o servidor receberá todas as informações. Se estas informações precisarem de alguma conversão, isso dependerá da lei da adequação de interface.

Com essas três leis pretendemos formalizar a metodologia para criação de ambientes com Interpercepção. Seguindo estas leis e o protocolo de conversão formalizado anteriormente é possível garantir que qualquer ambiente virtual possa ser transformado em um ambiente com Interpercepção.

Capítulo 6

Experimentos e Resultados

Para validar a teoria apresentada neste trabalho foram implementados dois estudos de caso. No primeiro estudo de caso a Interpercepção foi aplicada na construção de um ambiente virtual colaborativo chamado Garagem Virtual. No segundo estudo de caso aplicamos Interpercepção na construção de jogo multi-player *on-line* chamado Futebol de Botão Interp. Apresentamos neste capítulo os resultados alcançados nestes dois estudos de caso.

6.1 Garagem Virtual InterP

O Garagem Virtual é um ambiente virtual colaborativo para divulgação de músicas de bandas estreantes e amadoras. Este ambiente foi idealizado para a realização de festivais virtuais de música. O nome Garagem Virtual vem do fato que muitas bandas começam sua carreira ensaiando em garagens. O Festival Garagem Virtual foi proposto em uma parceira entre UFRN, UFPB e USP, tendo o apoio das Rádios Universitárias da UFRN e UFMA e TV Universitária da UFPB para a sua divulgação.

No Festival Garagem Virtual os artistas interessados se cadastram gratuitamente na página do festival preenchendo o formulário de inscrição. Ao se cadastrar, o artista escolhe um *login* e uma senha, e estes deverão ser usados para entrar na área de envio de seus trabalhos. Posteriormente é feita uma pré-seleção dos trabalhos remetidos. Esta seleção é feita por um grupo de jurados formado por músicos e críticos musicais que avaliam, individualmente, cada obra disponibilizada. Depois, as obras selecionadas são apresentadas ao público em um ambiente virtual.

O ambiente virtual do Festival Garagem foi desenvolvido originalmente com *interface* de visualização 2D. Este *interface* pode ser vista na figura 6.1. Esse ambiente possui um salão principal e cinco salas secundárias. Pelas salas secundárias vinte esculturas em forma de nota musical estão dispostas. Cada uma dessas esculturas está associada com alguma música pré-selecionada pelos jurados. Quando algum visitante se aproxima destas esculturas a música começa a ser executada na aplicação cliente deste visitante. É importante deixar claro que para cada interface de visualização está associada uma aplicação cliente diferente. Está aplicação é executada em forma de um *applet* Java.

Além de poder trafegar pelas salas do ambiente e ouvir as músicas, um usuário (visitante) também pode conversar com os demais usuários que estão trafegando pelo local

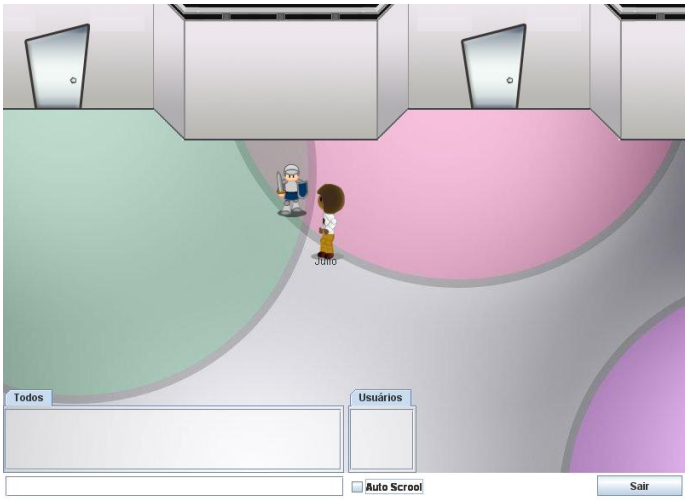

Figura 6.1: Interface de visualização do Festival Garagem

através de uma ferramenta de bate-papo que está acoplada a *interface* de visualização. Outra ação possível neste ambiente é possibilidade de votar na música que esta ouvindo.

Com base nas leis da Interpercepção foram desenvolvidos outras versões do ambiente Garagem: uma com *interface* de visualização 3D e outra com visualização 1D. Estes ambientes possuem a mesma modelagem. Mesmo que o ambiente 1D seja percebido apenas por informação textual, esta informação descreve o mesmo ambiente que é visto nas versões 2D e 3D.

Dentro do ambiente Garagem Virtual foram realizados três experimentos distintos. No primeiro experimento foram avaliadas as taxas de uso de memória e CPU de cada *interface* de visualização do Garagem. O objetivo deste teste é mostrar que uma máquina com poucos recursos de memória e CPU não será capaz de executar a *interface* de visualização 3D de forma satisfatória (ou mesmo a 2D), mas poderá executar as outras (ou apenas a 1D). Sendo assim, esse experimento mostrará que um usuário com tal máquina não poderia ter acesso ao ambiente Garagem, contudo, com uso da Interpercepção e a existência de outras *interfaces* de visualização, este usuário poderá optar por uma *interface* que se adapte à sua necessidade.

No segundo experimento foi avaliada a navegação dos avatares pelo ambiente. Neste experimento o objetivo era avaliar a capacidade da Interpercepção em transpor o movimento realizado por um avatar em uma *interface* de visualização para as demais. No caso específico da *interface* 1D, este movimento não é transposto, ele é na verdade simulado nas outras *interfaces*. Este experimento também validará as três leis da Interpercepção e o funcionamento do PORTAL no âmbito da conversão de mensagens, tanto de movimento quanto de adição de avatares.

No terceiro experimento testamos a colaboração no ambiente com o intuito de validar a lei da correspondência dos ambientes. Neste experimento avaliamos a capacidade de cada *interface* de visualização garantir o uso das mesmas atividades. Além disso testamos neste experimento a capacidade dos usuários em determinar de qual *interface* de visualização só demais usuários com os quais ele interagiu estão conectados.

6.1.1 Experimento de Taxas de uso do Sistema

Para realização deste experimento foram selecionadas duas máquinas com as configurações mostradas na tabela 6.1. Em cada uma dessas máquinas foi executada uma das *interfaces* de visualização em separado. Essa execução foi feita através de uma aplicação cliente, e outros 40 clientes acessaram o ambiente ao mesmo tempo, para simular uma situação usual de navegação em ambientes multiusuários. Em cada uma das execuções foram colhidas as taxas de uso da CPU e memória da *interfaces* de visualização executada para um destes clientes. Esses dados foram colhidos com a ferramenta JConsole que acompanha a distribuição Java 6 Development Kit [Sun 1990].

Requisitos	**Máquina 1**	**Máquina 2**
Processador	AMD Duron	Pentium 4
Freqüência do Processador	800 MHz	3.0 GHz
Memória	376 MB	2 GB
Placa de Vídeo	On-board	GForce
Memória de Vídeo	-	256 MB

Tabela 6.1: Resumo de máquinas usadas no experimento de taxas de uso

Para demonstrar a quantidade de recurso requerida pele *interface* de visualização 3D, temos na figura 6.2 uma comparação entre o desempenho da *interface* de visualização 3D na máquina 1 (sub-figura A) e na máquina 2 (sub-figura B).

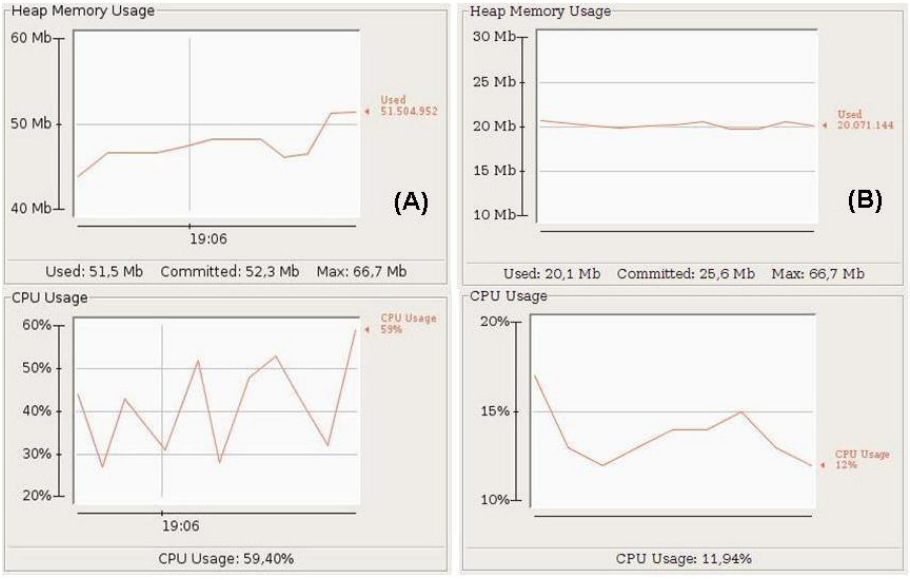

Figura 6.2: Comparação entre as Taxas de uso para a interface 3D: (A). Taxas da máquina 1, (B). Taxas da Máquina 2

Como mostra esta figura, a *interface* 3D ao ser executada na máquina 1 consome cerca de 50 MB da memória. O uso da CPU oscilou entre 30% e 60%. Esse uso elevado destes recursos pode inviabilizar a presença de um usuário no ambiente virtual aqui proposto.

Se este usuário usasse uma máquina semelhante ou de menor desempenho a esta, poderia não conseguir aproveitar o ambiente de forma satisfatória. Num ambiente sem Interpercepção, ele simplesmente desistiria de continuar acessando este ambiente. Contudo com a Interpercepção ele poderá optar por outra interface mais condizente com a realidade de sua máquina.

Se este usuário que acabou de desistir de usar *interface* de visualização 3D (porque sua máquina é semelhante à máquina 1) migrasse para a *interface* de visualização 2D, ele encontraria as taxas de uso apresentadas na figura 6.3.

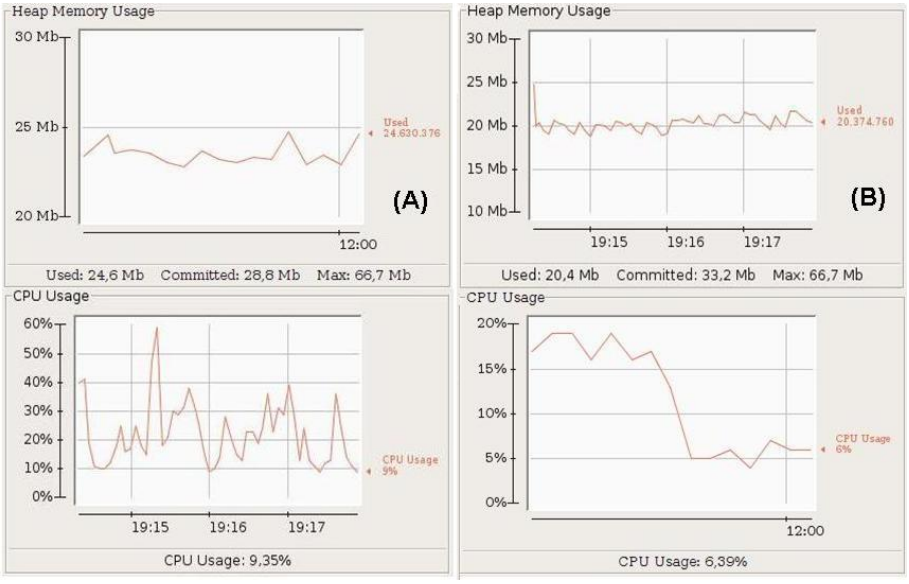

Figura 6.3: Comparação entre as Taxas de uso para a interface 2D: (A). Taxas da máquina 1, (B). Taxas da Máquina 2

Neste outra situação temos um desempenho mais vantajoso na máquina 1, que consegue executar a *interface* de visualização 2D com menos 25 MB da memória, apresentando uma redução de 50% se comparado com a mesma máquina na *interface* 3D. Já o uso da CPU oscilou entre 10% e 30%, que também demonstra uma redução de cerca de 50% no uso deste recurso. Essa redução torna mais viável para este usuário o uso da *interface* de visualização 2D. Portanto, neste caso ao invés de abandonar o ambiente o usuário precisou apenas migrar para um *interface* mais condizente com suas necessidades.

6.1.2 Experimento de Navegação

Como experimento para validar a navegação nos ambientes com Interpercepção, cinco usuários acessaram o ambiente. Dois deles estão conectados pela *interface* de visualização 3D, dois pela *interface* 2D e um pela interface 1D. Para tornar mais objetiva a explanação deste experimento, apresentamos na tabela 6.2 uma lista com o apelido (nick) de cada participante do teste, associados com a *interface* de visualização pela qual ele está acessando e o tipo do avatar que ele escolheu para representá-lo.

Apelido	Interface	Tipo do Avatar
Karla	1D	Gótica
Cas	2D	Dançarina
Claudio	3D	Black
Marco	3D	Punk
Julio	2D	Black

Tabela 6.2: Quadro de resumo do experimento Garagem

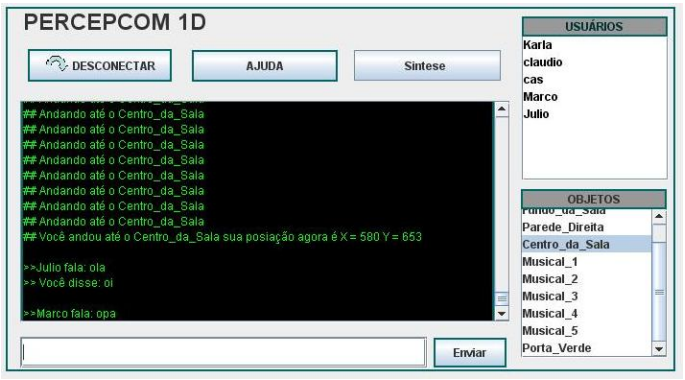

Figura 6.4: Percepção do usuário 1D no ambiente garagem

Neste experimento, os cinco usuários se reunirão na sala verde do ambiente Garagem. Os usuários Marco, Julio e Karla começam a conversar através da ferramenta de bate-papo. Esta situação é percebida pela usuária Karla da forma apresentada na figura 6.4. Já para o usuário Marco a situação descrita é percebida da forma apresentada na figura 6.5. Finalmente, o usuário Julio percebe o ambiente como mostrado na figura 6.6.

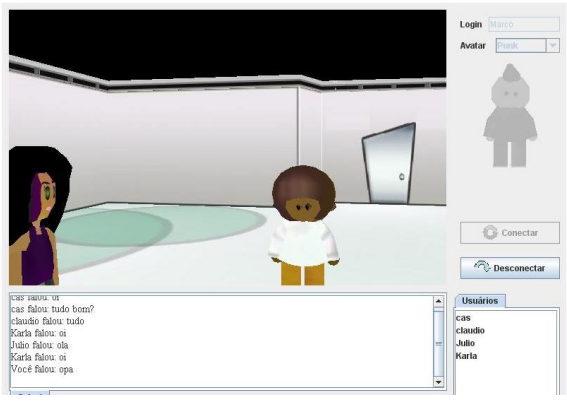

Figura 6.5: Visão do usuário Marco no experimento Garagem

Depois desta conversa o usuário Julio decide deixar a sala verde e voltar para a sala principal. Ao deslocar o seu avatar em direção à porta verde, Julio gera um fluxo de movimento que é percebido graficamente nas *interfaces* de visualização 2D e 3D, mas que não tem nenhum resultado visível no 1D, até que Julio deixe completamente a sala.

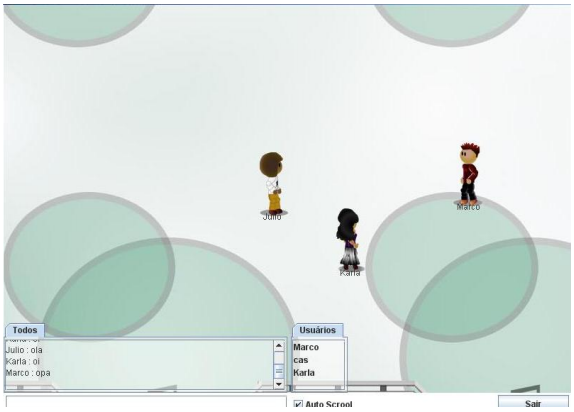

Figura 6.6: Visão do usuário Julio no experimento Garagem

O fluxo de movimento percebido pelo próprio Julio nessa ação esta demonstrado na figura 6.7. Nesta figura é mostrado uma seqüência de quadros que dispostos da esquerda para direita e de cima para baixo, mostram o avatar de Julio se movendo do local onde estava ocorrendo a conversa em direção a porta. No último quadro, o avatar já está no salão principal.

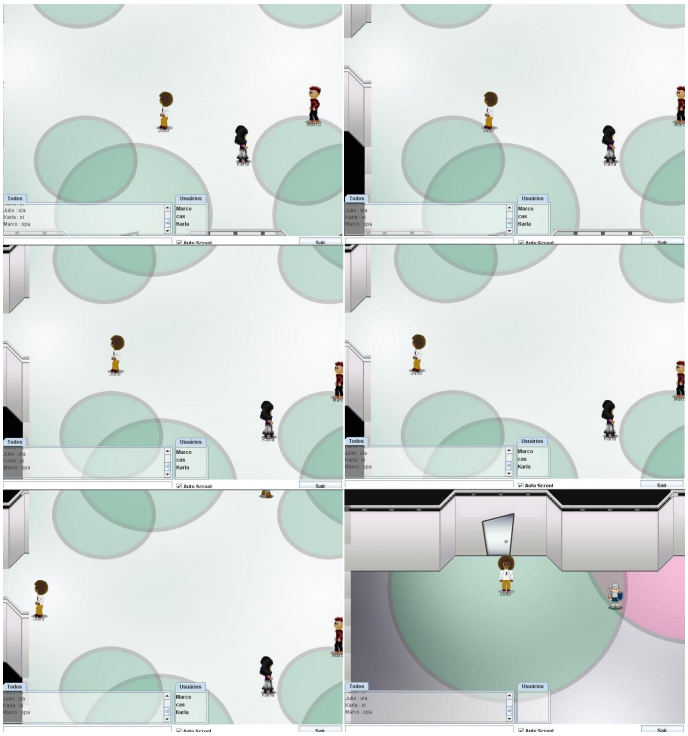

Figura 6.7: Resultado do experimento no 2D

A figura 6.8 mostra os resultados do experimento no 3D. Nos seis quadros dessa figura é possível ver o avatar de Julio se afastando da conversa e se aproximando da porta. Por fim no ultimo quadro, Julio não é mais visto na sala. Essa figura evidencia o funciona-

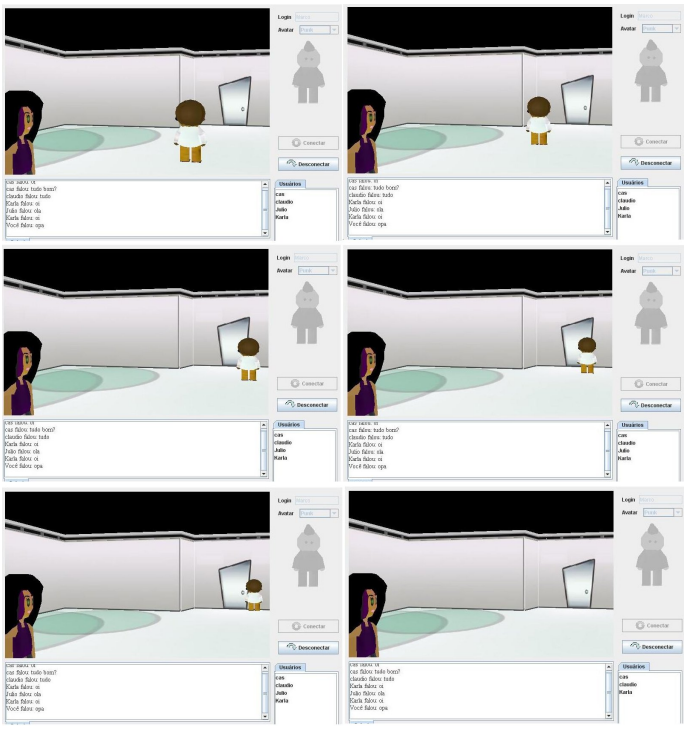

Figura 6.8: Resultado do experimento no 3D

mento do PORTAL na conversão das mensagens, à medida que mostra a atualização na *interface* 3D do movimento realizado por um usuário do 2D.

O movimento descrito por Julio gera várias mensagens de movimentos que são enviadas da sua aplicação cliente para o servidor. Estando no servidor as mensagens são então encaminhadas para os demais clientes conectados. O PORTAL ligado à aplicação cliente executada por Claudio filtra essa mensagem de movimento e descobre que ele é oriunda de um cliente 2D. Este PORTAL efetuara então a conversão dos pontos contidos nesta mensagem (pontos 2D) para pontos condizentes com a *interface* do seu cliente (pontos 3D). Além disso, é preciso ressaltar que o avatar é de uma dimensão diferente, contudo ele já foi adequado anteriormente quando Julio acessou esta sala. Neste ponto ele ainda permanece adequando, sendo representado em 3D na *interface* da aplicação cliente de Claudio. Quando Julio alcançar o último ponto dentro desta sala, a representação de seu avatar será removida da aplicação cliente de Claudio, o que resulta no desaparecimento do avatar de Julio da *interface* de visualização de Claudio, como mostrado no ultimo quadro, da figura 6.8.

Os resultados alcançados com este teste comprovaram a eficiência do PORTAL na conversão das mensagens, evidenciado pelo movimento dos avatares sendo atualizado em cada uma das interfaces. O experimento também comprovou a eficácia do servidor unificado para o tratamento das mensagens de todos os tipos de clientes, uma vez que todos conseguiram receber as mensagens de texto e movimento dos demais usuários. As figuras também evidenciaram a correspondência dos ambientes, ou seja, cada usuário estava percebendo a mesma sala, os mesmos objetos e as mesmas ações.

6.1.3 Experimento de Colaboração

Para realizar este experimento deixamos o ambiente Garagem aberto a visitas externas durante dois meses. Ao final destes dois meses avaliamos a quantidade de usuários que acessaram o ambiente através de cada interface. O resultado foi:

- A *interface* 2D foi acessada 115 vezes;
- A *interface* 3D foi acessada 74 vezes;
- A *interface* 1D foi acessada 52 vezes;

Para nós ficou claro que os usuários tinham certa preferência pela *interface* 2D, provavelmente por causa do seu modo de navegação mais intuitivo e pelo fato da mesma não exigir tantos recursos. A *interface* de visualização 3D além de requerer mais recursos de CPU e memória (como mostrado no experimento 1 deste estudo de caso), ele também requer a instalação de um *Plugin* o que já afasta alguns usuários. O uso mínimo da *interface* 1D é decorrente da sua *interface* de visualização não ser muito amigável. Contudo, era nosso intuito que ela realmente não fosse amigável, que foi o mais fiel possível aos **MUD**'s.

Ainda realizamos a seleção de um grupo de 3 usuários de cada uma das interfaces para avaliar se conseguiram utilizar as ferramentas de colaboração do ambiente e como os mesmos se comportaram nele. O resultado desta avaliação está na Tabela 6.3.

Na tabela 6.3 a coluna 1D representa a interação do usuário da linha em questão com outros usuários da *interface* de visualização 1D. O mesmo vale para as colunas 2D e 3D. A coluna Salas define a quantidade de salas que o usuário da linha em questão visitou. A coluna Tempo define a quantidade de tempo que o usuário da linha em questão passou no ambiente durante a sua visita. A coluna Conversação define a quantidade de linhas de texto que aquele usuário conversou durante a sua visita. Por último, a coluna Transparência define se o usuário interagiu sem perceber a *interface* de origem dos outros usuários com quem ele estava interagindo.

Interface	1D	2D	3D	Salas	Tempo (s)	Conversação (linhas)	Transparência
3D	2	2	3	2	3931	2	Sim
3D	3	1	0	1	168	3	Sim
3D	4	0	0	3	1009	59	Sim
2D	2	1	1	5	145	1	Sim
2D	4	6	0	1	514	10	Sim
2D	4	2	7	2	1052	24	Sim
1D	5	1	1	2	900	1	Sim
1D	0	3	3	1	274	1	Sim
1D	2	0	9	1	939	17	Sim

Tabela 6.3: Resultado dos teste de Colaboração

Os resultados nesta tabela evidenciam que a maioria dos usuários interagiu com outros conectados por *interfaces* distintas. Todos os usuários aqui selecionados conseguiram

utilizar a ferramenta de conversação. Em média, os usuários do 1D foram os que conversarão menos e isto pode ser explicado pelo fato que todos as interações neste *interface* se dão através do teclado, e não apenas a conversação.

Os usuários do 2D foram os que me média visitaram mais salas. Isto pode ser respondido pela forma de navegação neste ambiente ser através do mouse e visualização do ambiente ser em terceira pessoa. Na *interface* 3D como a navegação é feita pelos botões direcionais do teclado e a visão é em primeira pessoa, alguns usuários podem sentir certa dificuldade para trafegar pelo ambiente. O número de salas visitadas foi menor no 1D e isto pode ser explicado pelo tempo perdido pelos usuários digitando os comandos para trafegar pelas salas. Também pode ser levado em consideração a falta de atrativos visuais desta *interface*.

O Tempo gasto pelos usuários foi maior no 3D, seguido do 1D e depois do 2D. Isso pode ser explicado pelo fato de que a *interface* 2D é a de navegação mais fácil e por isso os usuários conseguem visitar todo o ambiente sem perder muito tempo, tendo em vista que foram os usuários desta *interface* que visitaram mais salas.

Para medir a transparência da colaboração, ao final da navegação o usuário era questionado se ele havia notada em qual *interface* estava os demais usuários com os quais eles estavam interagindo. Todos os usuários desta amostragem alegaram que os demais usuários estavam acessando pela mesma *interface* que eles. Como todos os acessos foram formados por grupos mistos de usuários, atestamos a Transparência em todas as interações, ou seja, nenhum usuário percebeu de fato em qual *interface* de visualização estavam os demais.

6.2 Futebol de Botão InterP

O foco central deste experimento está na aplicação do conceito de Interpercepção para construção de ambientes virtuais compartilhados. Contudo, decidimos estender a sua aplicação para construção de jogos *interperceptivos*. O futebol de botão InterP é a primeira experiência na construção de jogos segundo os preceitos da Interpercepção.

Neste jogo, dois jogadores disputam uma partida de futebol de botão pela *web*. O diferencial deste jogo é que os usuários podem estar conectados por *interfaces* de visualização diferentes. Mesmo estando em *interfaces* distintas, os jogadores compartilham o mesmo ambiente de jogo. Além disso, este jogo possui um modo *Torcida*, onde outros usuários podem acessar os sistema apenas para assistir ao jogo.

No ambiente de jogo criado para essa aplicação foi utilizada apenas as *interfaces* 2D e 3D. No experimento realizado, um jogador acessou o jogo através da *interface* 2D e escolheu o time do Palmeiras para jogar. O segundo jogador escolheu o time do São Paulo e acessou o ambiente através da interface 3D. Para demonstrar a interpercepção do jogo, a figura 6.9 mostra a visão do jogador que está no 2D. Já a figura 6.10 mostra a visão do jogador no 3D.

Como demonstram as figuras 6.9 e 6.10, a visão do jogador no 3D é mais abrangente do que a visão do jogador no 2D. Isto pode conceder ao jogador que está no 3D certa vantagem estratégica, contudo não faz parte das preocupações deste trabalho o estudo da justiça de jogo envolvida em jogos Interperceptivos. Nosso interesse é demonstrar

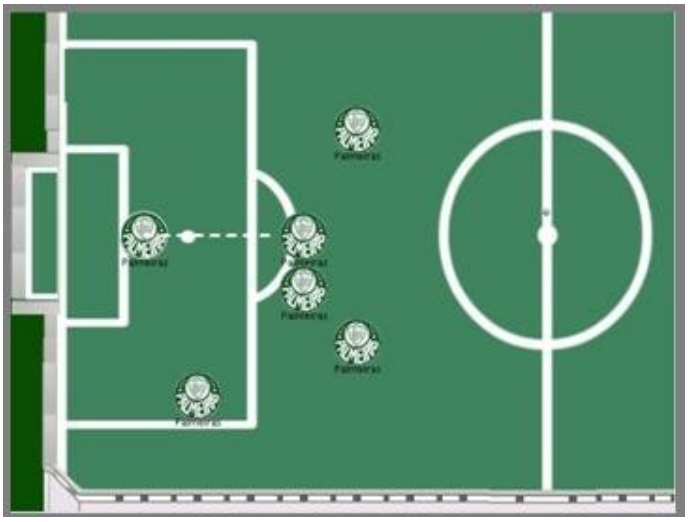

Figura 6.9: Jogo InterP na interface 2D

a viabilidade de construir jogos que sigam as leis da Interpercepção, como o jogo aqui demonstrado.

Dando continuidade ao experimento do Futebol de botão InterP, apresentamos na figura 6.11 a visão do torcedor na *interface* de visualização 2D. Nesta figura, uma usuária, chamada Josefina, com o avatar do tipo Gótica faz algum comentário sobre o jogo. Este comentário é percebido por todos os demais usuários presentes na torcida, inclusive o usuário Paulo que está com o avatar do tipo Punk. Paulo está conectado através da interface 3D e sua visão da cena pode ser vista na figura 6.12. Como mostra esta figura, Paulo está vendo a mensagem enviada por Josefina. Na visão de Paulo, Josefina, o ambiente de jogo e os demais usuários são todos 3D.

Figura 6.10: Jogo InterP na interface 3D

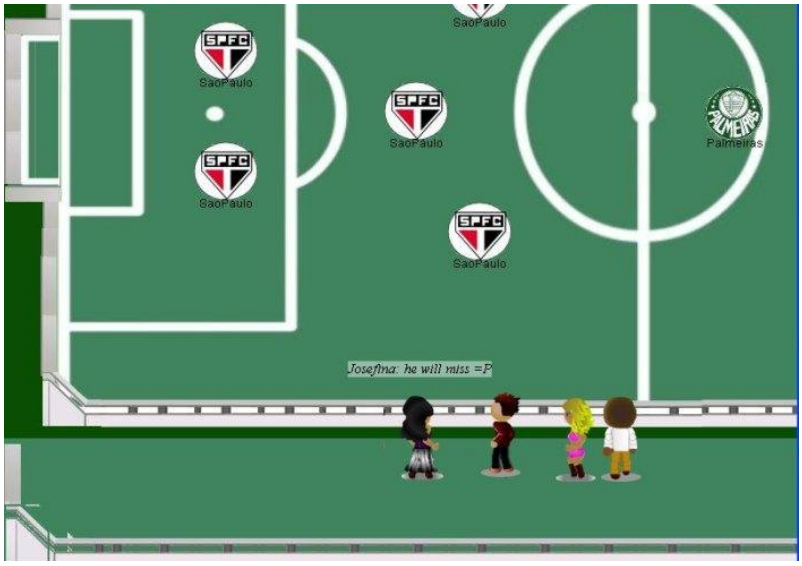

Figura 6.11: Visão do modo Torcida no 2D

Figura 6.12: Visão do modo Torcida no 3D

Capítulo 7

Conclusão e Perspectivas

Neste trabalho, propomos o paradigma da Interpercepção, uma abordagem para a interligação de ambientes virtuais multiusuários com *interfaces* de visualização distintas. Usamos o modelo de desenvolvimento baseado em prototipagem, sendo desenvolvido um protótipo inicial, que foi melhorado no decorrer desta pesquisa. Usamos os padrões de projeto **MVC** (*Model - View - Controller*) e *Strategy* para a modelagem arquitetural do sistema desenvolvido, sobre a plataforma JUDE [JUDE 2007] e notações UML. Como ferramenta de edição e atualização de código, foi utilizado o Eclipse (usando a técnica CVS).

A motivação e ponto de partida para o desenvolvimento das teorias introduzidas foi o sistema *PercepCom* [Dantas et al. 2005], por nós desenvolvido em trabalhos anteriores, que surgiu, inicialmente, como um componente gráfico independente de plataforma para a criação de ambientes virtuais tridimensionais. A implementação original do *PercepCom* foi melhorada, motivados pela necessidade de criação de um sistema portável com suporte a vários usuários em um ambiente virtual colaborativo através da web. Foi idealizado um ambiente que utilizasse também uma *interface* de visualização 2D, além da 3D. Mesmo que uma gama alta de clientes fosse alcançada com estas duas versões do *PercepCom*, foi ainda proposta uma terceira *interface* de visualização, puramente textual, inspirada nos **MUD**'s, nascendo assim o *PercepCom*1D. Esta versão textual foi também proposta para integração de usuários portadores de deficiência visual e conta com ferramentas de síntese e reconhecimento de voz. Assim, fechamos nossa proposta, com as correntes implementações e experimentos realizados, mostrando sua versatilidade.

Podemos considerar como principais contribuições do presente trabalho o protocolo de comunicação usando diferentes interfaces, propriamente dito, que permite a possibilidade de usuários com interfaces distintas ficarem no mesmo nível de abstração e de compartilhamento de informações em ambientes virtuais. Ainda, podemos citar as técnicas e algoritmos para conversão entre as interfaces distintas, e a arquitetura cliente-servidor (baseado no H-N2N) do sistema usado nas aplicações, que pode ser facilmente adaptada para outras aplicações (portanto, uma base para a construção de um *framework* mais geral). Ressaltamos a possibilidade de todos os usuários conectados poderem interagir mutuamente.

Sobre a arquitetura proposta neste trabalho podemos desenvolver aplicações que podem ser usadas por usuários que possuem computadores com alto poder de processamento conectados a uma rede de alta velocidade, bem como também podemos ter usuários uti-

lizando computadores antigos, com uma configuração de *hardware* e *software* obsoletas, conectados através de uma conexão discada, de maneira transparente. O resultado é um ambiente virtual que disponibiliza diferentes *interfaces* de visualização para que seus usuários o acessem, visando garantir uma maior acessibilidade, principalmente para os usuários com poucos recursos. Usuários com equipamentos de baixo desempenho que não conseguiriam acessar o ambiente através de uma *interface* de visualização 3D, podem optar por *interfaces* de visualização menos exigentes, como a *interface* de visualização 2D. Já usuários com deficiência visual, por exemplo, poderiam, através das técnicas de síntese e reconhecimento de voz, utilizar o ambiente através da *interface* de visualização textual. Através da Interpercepção mostramos que é póssivel interligar ambientes com *interfaces* 1D (textual), 2D e 3D, formando um único ambiente compartilhado por todos os usuários.

Os sistemas que utilizam a interpercepção demonstram potencial para serem usados como uma importante ferramenta de inclusão digital não só para a população de baixa-renda, mas também para deficientes visuais e auditivos. Eles permitem o acesso a um público maior de usuários que em muitos casos não teriam acesso a uma variedade de sistemas se não dispusessem de computadores com a configuração mínima para executá-los. Mediante esta capacidade de abranger uma quantidade maior de público, um sistema interperceptivo abre novas perspectivas de negócios que antes só podiam existir para uma quantidade mais restrita de consumidores.

Nos experimentos e testes, mostramos a aplicação do paradigma proposto de interpercepção na construção de um ambiente virtual multiusuário e de um jogo *on-line*. Em ambos os estudos de caso, os resultados alcançados demonstram a viabilidade do uso do modelo proposto, visando melhorar o atual paradigma de inclusão social e digital. Contudo, percebemos que é possível nem todos os sistemas possuam um escopo que possa permitir a utilização da interpercepção. Alguns estilos de jogos, como por exemplo, jogos de tiro em primeira pessoa, possuem em sua natureza elementos que talvez não possam ser transportados para dimensões diferentes. No caso desses jogos, seria injusto um jogador na interface 2D poder mirar em um jogador na interface 3D. Do mesmo modo, ainda não estudamos a possibilidade de transpor esse estilo de jogo para 1D. Convém ressaltar que este não foi o escopo do presente trabalho, não tendo sido realizado um estudo de técnicas para solucionar esse e outros problemas ligados à justiça de jogo (*feernees*). Outros desenvolvimentos futuros são necessários, visando introduzir estas e outras melhorias na arquitetura proposta.

7.1 Perspectivas

A Interpercepção abre margem para muitas outros estudos. O primeiro, apontado anteriormente, é o estudo da justiça de jogo em jogos interperceptivos. O desenvolvimento de jogos desse tipo surge como um novo paradigma na área de desenvolvimento de jogos. Contudo, o sucesso deste tipo de jogo está ligado a um estudo mais detalhado das técnicas para torna-lo equilibrado para todos os participantes.

A Interpercepção abordada neste trabalho diz repeito ao seu uso em aplicações para computadores pessoais. Contudo, a Interpercepção pode ser estendida para criação de

ambientes virtuais e jogos para celular. Estes ambientes e jogos, através da Interpercepção poderiam se comunicar com outras versões destas aplicações que sejam executadas através de um computador pessoal. Deste modo, essa estensão da Interpecepção vai além da diferença do tipo da *interface* de visualização, abrangendo também a diferença de dispositivo. No escopo dos diferentes dispositivos que poderiam ser interligados por Interpercepção, podemos incluir também a TV digital. Através desta extensão, o mesmo ambiente poderia ser acessado por celular, por computador pessoal, ou pela TV Digial, criando, portanto, um ambiente pervasivo e de Interpercepção.

Na definição das leis da Interpecepção, definimos que o ambiente visualizado deve ser o mesmo em todos as *interfaces*. Deste modo, é possível converter os avatares de uma *interface* de visualização para a outra, sem muito transtorno. Mas, se os avatares fossem convertidos de um outro ambiente completamente diferente as regras de conversão aqui definidas não seriam válidas. Assim, estudos e desenvolvimentos visando resolver este problema também podem ser apontados como desenvolvimentos futuros, visando melhorar a presente proposta. No caso, o uso e adoção de um padrão para a modelagem dos avatares pode ser um dos caminhos escolhidos.

Referências Bibliográficas

ActiveWorlds (2007), 'The active worlds website', http://www.activeworlds.com.

Babski, Christian e Stéphane Carion (2003), knowscape - a collective knowledg architecture, *em* 'Proceedings of the Virtual Environments, Human-Computer Interfaces, and Measurement Systems', pp. 27–29.

Bartle, Richard (1990), 'Early mud history', Article posted to USENET group rec.games.mud.

Benford, Steve e Chris Greenhalgh (2001), 'Collaborative virtual environments', *Communications of the ACM* **44**(4), 79.

Branco, António, António Teixeira, Ana Tomé e Francisco Vaz (1997), Um sintetizador articulatório para o estudo da produção da voz, *em* 'Actas da Conferência Nacional de Telecomunicações', pp. 405–407.

Brown, Barry, Ian MacColl, Matthew Chalmers, Areti Galani, Cliff Randell e Anthony Steed (2003), Lessons from the lighthouse: Collaboration in a shared mixed reality system, *em* 'Proceedings of the CHI 2003 Conference on Human Factors in Computing Systems', pp. 577–584.

Burbec, Steve (1987), 'Applications programming in smalltalk-80:how to use model-view-controller'.

Burlamaqui, Aquiles, Guido Lemos, Jauvane Oliveira, Marlos Oliveira e Luiz Marcos G. Gonçalves (2006), H-n2n - uma solução escalável para ambientes virtuais colaborativos massivos, *em* 'XII Simpósio Brasileiro de Sistemas Multimídia e Web'.

Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad e Michael Stal (1996), *Pattern-oriented software architecture: a system of patterns*, 1ª edição, John Wiley and Sons.

Damer, Bruce (1997), *Avatars: Exploring and Building Virtual Worlds on the Internet*, 1ª edição, Peachpit Press.

Dantas, Rummenigge R., Cláudio Schneider, Josivan Xavier, Aquiles Burlamaqui e Luiz Marcos G. Gonçalves (2005), Um componente gráfico para o desenvolvimento de ambientes virtuais massivos com java3d, *em* 'XI Simpósio Brasileiro de Sistemas Multimídia e Web'.

Dave Crane, Eric Pascarello (2006), *Ajax in Action*, Manning, New York, EUA.

Dieterle, Edward e Jody Clarke (2007), Multi-user virtual environments for teaching and learning, *em* 'Encyclopedia of multimedia technology and networking'.

Dix, Alan, Janet E. Finlay e Gregory D. Abowd (2004), *Human-Computer Interaction*, 3ª edição, Pearson.

Eclipse, Fundation (2007), 'Eclipse.org home', http://www.eclipse.org/.

Ellis, Steve R. (1994), 'What are virtual environments?', *IEEE Computer Graphics e Applications* **14**(1), 17–22.

Elllis, Steve (1991), Nature and origin of virtual environments: a bibliographical essay, *em* 'Computer Systems in Engineering', pp. 321–346.

Foley, James D., Andries van Dam, Steven K. Feiner e John F. Hughes (1995), *Computer Graphics: Principles and Practice in C*, 2ª edição, Addison-Wesley Professional.

Gamma, Erich, Richard Helm e Ralph Jonhson (1999), *Padrões de Projeto*, 1ª edição, Bookman, São Paulo, Brasil.

Google (2007), 'Orkut', http://www.orkut.com.

Gravity, Interactive (2007), 'Ragnarok on-line', http://iro.ragnarokonline.com/.

Greenhalgh, Chris (1999), *Large Scale Collaborative Virtual Environments*, Springer.

Gygax, Gary (1987), *Roleplaying Mastery*, Perigee, New York, EUA.

Han, JungHyun, Hoh Peter In e Jong-Sik Woo (2005), Towards situation-aware cross-platform ubi-game development, *em* 'Proceedings of Pervasive2005', pp. 8–13.

Jones, Randolph M. (2000), 'Design and implementation of computer games: a capstone course for undergraduate computer science education', *ACM SIGCSE Bulletin* **32**(1), 260–264.

JUDE, Community (2007), 'Uml modeling tool'.

Morningstar, Chip e Randy Farmer (1991), The lessons of lucasfilm's habitat, *em* M.Benedikt, ed., 'Proceedings of Cyberspace: First Steps', pp. 273–302.

MUD, TOP (2007), 'Rankings of muds', http://www.topmudsites.com/.

Paulino, Julio C., Rummenigge R. Dantas, Claudio Schneider, Samuel Azevedo, Aquiles Burlamaqui, Guido Lemos e Luiz Marcos G. Gonçalves (2006), Percepcom2d, uma abordagem 2d para ambientes virtuais colaborativos multi- usuário, *em* 'XII Simpósio Brasileiro de Sistemas Multimídia e Web'.

Preiser, Wolfgang F. E. e Elaine Ostroff (2001), *Universal Design Handbook*, 1ª edição, McGraw-Hill Professional.

Reis, Eduardo (2007), 'Hero rpg quest', http://www.herorpgquest.hpg.ig.com.br/.

Schollmeier, Rüdiger (2002), 'A definition of peer-to-peer networking for the classification of peer-to-peer architetures and applications', *IEEE Internet Computing* .

Schröder, Marc (2001), Emotional speech synthesis: A review, *em* '7th European Conference on Speech Communication and Technology', pp. 561–564.

Sebrechts, Marc M., Joanna Vasilakis, Michael S. Miller, John V. Cugini e Sharon J. Laskowski (1999), Visualization of search results: A comparative evaluation of text, 2d, and 3d interfaces, *em* 'Proceedings of the 22nd Annual Int'l ACM-SIGIR Conf. Research and Development in Information Retrieval'.

Seven, Hive (2007), 'Social web games', www.hive7.com.

Sproat, Richard (1997), *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*, 1ª edição, Springer.

Stephenson, Neal (2007), *Snow Crash (SFBC 50th Anniversary Collection)*, SFBC.

Sun, Microsystens (1990), 'Java platform, standard edition 6 release', http://java.sun.com/javase/6/.

Sutherland, Ivan (1965), The ultimate display, New York, EUA, pp. 506–508.

Sutherland, Ivan (1968), A head-mounted three dimensional display, *em* 'Fall Joint Computer Conference', Washington, EUA, pp. 757–764.

Tanenbaum, Andrew S. e Maarten van Steen (2001), *Distributed Systems: Principles and Paradigms*, 1ª edição, Prentice-Hall.

Tanenbaum, Andrew S. e Maarten van Steen (2006), *Distributed Systems: Principals and Paradigms*, 2ª edição, Prentice-Hall International.

West, Tracey (2007), *RuneScape: The Official Handbook*, 1ª edição, Scholastic Library Publishing.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo