UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE GRADUAÇÃO

Jogos Multiplataforma Multiusuário

Davi de Vasconcelos Pedrosa

Orientador: Geber Lisboa Ramalho
Co-orientador: Carlos André Guimarães Ferraz

Recife
Outubro de 2006

TRABALHO DE GRADUAÇÃO

Jogos Multiplataforma Multiusuário

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Davi de Vasconcelos Pedrosa

Orientador: Geber Lisboa Ramalho
Co-orientador: Carlos André Guimarães Ferraz

Recife
Outubro de 2006

Agradecimentos

A minha família, pelo amor e apoio.

À minha querida Flávia, pelo carinho e incentivo.

A Trinta, Geber e Carlos, pelo suporte, orientação e confiança.

Resumo

Jogos multiusuário *online* despertam cada vez mais interesse comercial e acadêmico, e têm evoluído bastante nos últimos anos. Atualmente, exemplos desenvolvidos para telefones celulares e outros dispositivos móveis já oferecem novas experiências de jogo ao usuário, com características de pervasividade. Jogos multiplataforma multiusuário uniriam as vantagens de jogabilidade de diferentes plataformas, pois poderiam ser acessados a partir de dispositivos distintos, como telefones celulares e *desktops*. Contudo, a implementação deste tipo de jogo é bastante custosa, por ser necessária a adaptação da jogabilidade para cada jogador, a depender do dispositivo utilizado em questão. Partindo de um *middleware* para jogos multiplataforma multiusuário em desenvolvimento, este trabalho propõe a especificação e implementação de dois de seus serviços. Estes serviços adaptam, respectivamente, a percepção e a interação dos jogadores com o jogo, dependendo do dispositivo utilizado. Como estudo de caso foi também criado um conjunto de cenários de jogos para validação dos serviços.

Palavras-chave: Jogos multiusuário, jogos pervasivos, *middlewares* para jogos, ciência de contexto.

Sumário

Lista de figuras

CAPÍTULO 1

Introdução

Nos últimos anos, tem ocorrido um grande desenvolvimento na área de jogos multiusuário *online* – jogos em que várias pessoas participam utilizando computadores distintos conectados através de uma infra-estrutura de comunicação. Tal progresso deve-se principalmente ao avanço das tecnologias de comunicação e ao sucesso da Internet, despertando inclusive um crescente interesse comercial nestas aplicações. Como exemplo, em 2005 o mercado de jogos multitusuário *online* movimentou cerca de 3,4 bilhões de dólares [Ign06].

Enquanto isso, também evolui e populariza-se cada vez mais a computação móvel. A inerente conectividade dos dispositivos móveis, somada ao surgimento de plataformas de desenvolvimento como Java ME [SM06] e BREW [QI06] resultam em uma oferta cada vez maior de serviços para esses dispositivos. Nos últimos anos, o aparecimento de jogos multiusuário *online* para telefones celulares evidencia o potencial da computação móvel para a área de entretenimento digital.

Este potencial se traduz em uma grande mudança na maneira como um jogador utilizando um dispositivo móvel se relaciona com um jogo. Uma das possíveis abordagens permite ao jogador permanecer sempre conectado ao jogo a partir de seu dispositivo. É uma situação que se contrapõe à tradicional idéia de jogos multiusuário, na qual jogadores possuem máquinas estacionárias com grande poder computacional e dispondo geralmente de uma conexão de alta velocidade.

Atualmente, já são vários os jogos existentes, comerciais inclusive, que exploram individualmente estes cenários, cada um dos quais composto por diferentes tipos de

plataformas[1]. Sendo assim, um jogo específico pode oferecer aos usuários características interessantes da plataforma para o qual foi desenvolvido: mobilidade, para o caso de jogos multiusuário móveis, ou uma interface gráfica rica e altamente interativa, para os jogos de computadores *desktop* ou consoles de *videogame*. Desta forma, pode-se imaginar um cenário ainda mais inovador, onde um mesmo jogo seria utilizado a partir de plataformas distintas. Com cada uma delas provendo uma experiência de jogo própria ao usuário, o jogo poderia ser encarado pelos jogadores de diferentes maneiras, dependendo de como eles o estivessem acessando.

Deste modo, vislumbra-se o conceito de computação pervasiva[2], considerada uma evolução ao atual momento da computação móvel. No entanto, a infra-estrutura sugerida neste trabalho para a incorporação de pervasividade é aquela já presente e disponível por operadoras de telefonia que oferecem acesso à Internet. Fica restrito, portanto, o ambiente de computação pervasiva a ser utilizado, haja vista a inviabilidade de todas as possibilidades que a mesma oferece [CDK05a] [oST00].

Neste sentido, as características pervasivas a serem abordadas são *(i)* permitir que os usuários possam ter acesso à informação, de qualquer local, podendo utilizar diferentes dispositivos e suas possibilidades de conexão, e *(ii)* adaptar a informação recebida ou enviada pelos dispositivos de acordo com o contexto do jogador.

As diferenças existentes nos dispositivos em termos de usabilidade, interfaces de apresentação, características de comunicação e capacidade de processamento exige que as interações e as formas de apresentação do jogo sejam adaptadas às características do dispositivo utilizado pelo jogador em um dado instante. Desta maneira, seria extremamente custoso para desenvolvedores a implementação de um jogo multiplataforma multiusuário.

A solução mais viável é a criação de uma infra-estrutura que amenize o esforço requerido para a implementação destes jogos, reduzindo seus custos e tempo de desenvolvimento. Atualmente, já existem diversos *middlewares* [Ber96] [Emm00] [CDK05b] específicos para jogos, que geralmente facilitam a manutenção de estado compartilhado e comunicação entre jogadores. Um exemplo consiste no projeto MMMOG [SAP+05],

[1] O termo plataforma possui, neste trabalho, a idéia de um ambiente de execução de software de um dispositivo específico, baseado em um sistema operacional e uma tecnologia de hardware.
[2] Embora não exista em português o termo *pervasivo*, muitas pesquisas no Brasil utilizam este termo como uma tradução do termo em inglês *pervasive*, que significa "difundido inteiramente por toda parte" [MWO06].

desenvolvido em conjunto com o Centro de Informática da UFPE, o C.E.S.A.R, a Meantime e a Unicamp, e que foca em jogos massivos para celular.

Ainda no Centro de Informática, é desenvolvida uma tese de doutorado cujo objetivo é a implementação de um *middleware* para jogos pervasivos massivos multiusuário multiplataforma (*Pervasive Massive Multiplayer Multiplatform Game* – PM³G) [Tri06a]. Um PM³G, como o próprio nome diz, trata-se de um jogo multiplataforma multiusuário, ou seja, um jogo multiusuário *online* que permite acesso de forma coerente a partir de diferentes dispositivos usados, apesar da grande diferença existente entre seus contextos. Além disso, um PM³G também aproveita situações e serviços específicos do cenário móvel para permitir interações mais intensas entre jogadores móveis, como a formação de redes espontâneas através de diferentes tecnologias de comunicação e o uso de recursos baseados na localização do usuário (*Location Based Services* – LBS).

1.1 Objetivo

Neste trabalho são detalhadas a especificação e a implementação de uma parte do *middleware* proposto na tese de doutorado mencionada anteriormente. Em primeiro lugar, é especificado um modelo de suporte a jogos multiplataforma através do conceito de *middleware*, permitindo a adaptação da jogabilidade em relação ao contexto do jogador e a integração de diferentes dispositivos. Para tanto, dois serviços básicos foram criados: o serviço de adaptação de conteúdo e o de adaptação de interação.

Também é detalhada a implementação do modelo especificado, possibilitando a criação de jogos com características multiplataforma. Estes experimentos baseiam-se no *framework* em desenvolvimento na tese de doutorado motivadora deste trabalho, sendo desenvolvido utilizado-se a plataforma Java e o perfil Java Platform, Standard Edition (Java SE) [SM06b].

Finalmente, é feito o estudo de caso através da criação de cenários de jogo simples, utilizando os serviços implementados para validação tanto do modelo quanto da implementação.

1.2 Visão geral

Os capítulos 2 e 3 contextualizam o assunto tratado neste trabalho. O primeiro aborda jogos multiusuário de uma maneira geral. São mostrados os principais estilos e cenários, são discutidos aspectos técnicos relacionados à sua implementação e também são apresentados *middlewares* relacionados a estes jogos. O capítulo 3 trata de jogos multiplataforma multiusuário, citando exemplos de iniciativas comerciais e acadêmicas na área, além de introduzir e descrever o *middleware* PM³G.

O capítulo 4 consiste da principal contribuição deste trabalho, e nele são detalhadas a especificação e a implementação do Serviço de Adaptação de Conteúdo e de Adaptação de Interação. No capítulo 5 são descritos os cenários de jogos criados para validar os serviços implementados. Finalmente, no capítulo 6 são apresentadas as conclusões acerca do trabalho desenvolvido.

CAPÍTULO 2

Jogos multiusuário

Jogos multiusuário são simulações de ambientes onde cada jogador busca atingir um certo objetivo através da interação com outros jogadores e com o ambiente [Cec05]. Este ambiente representa o mundo virtual, o espaço compartilhado entre jogadores onde o jogo é realizado.

Este mundo virtual tipicamente constitui-se do mapa do jogo, personagens controlados por diferentes jogadores, objetos mutáveis (alimentos, ferramentas, armas, dentre outros), além de personagens controlados pelo computador, chamados de *Non-player Characters* (NPC). A representação de cada jogador no mundo virtual é chamada *avatar*, termo comum às aplicações de realidade virtual. Cada elemento do jogo possui um estado associado, composto por atributos como sua posição no mundo virtual e um conjunto de propriedades específicas de cada jogo, como habilidades, possessões ou energia vital. O conjunto dos estados de cada elemento em um dado momento reflete o estado global do jogo. Este estado é visualizado, compartilhado e modificado de acordo com ações dos jogadores presentes no jogo. Em geral, um jogador pode realizar três tipos de ações: a mudança de posição de seu(s) avatar(es), interações com objetos do mundo virtual e interações com outros jogadores, estando cada uma destas ações sujeita às regras determinadas pelo projeto do jogo (game design).

No entanto, mesmo classificados como aplicações de espaço compartilhado, jogos multiusuário apresentam características peculiares. Primeiramente, seu foco está na competição, e não apenas a colaboração, entre seus usuários. Esta característica influencia diretamente um segundo aspecto, que jogos multiusuário devem ser tolerantes à trapaça [Cec05]. Enquanto os outros ambientes de espaço compartilhado geralmente utilizam redes privadas, com participantes confiáveis, jogos multiusuário utilizam redes públicas, onde

certos jogadores fazem uso de meio anti-éticos para obtenção de vantagens ilegais em relação a jogadores honestos.

Neste capítulo são abordados alguns aspectos de jogos multiusuário, importantes para uma melhor compreensão do restante do trabalho. Primeiramente, fala-se a respeito de suas principais classificações e de seus cenários atuais. Em seguida, são detalhados alguns aspectos técnicos e, por fim, é introduzida a idéia de *middlewares* para jogos multiusuário.

2.1 Estilos de jogos multiusuário

Jogos multiusuário podem ser classificados sob diferentes aspectos. Em relação ao andamento da simulação, um jogo de tempo real é aquele onde o jogador envia comandos de forma independente à passagem do tempo da simulação [Cec05]. Em outras palavras, os jogadores realizam ações sem uma ordem pré-definida, enviando dados continuamente e modificando concorrentemente o estado do jogo. Em um jogo baseado em turnos, por sua vez, cada participante tem bem definido o seu momento de realizar uma jogada, havendo uma alternância entre as ações de cada jogador. Um exemplo é o jogo de xadrez, no qual a partida não avança até que o jogador da vez realize a sua jogada.

Jogos multiusuário de tempo real geralmente impõem restrições quanto ao atraso máximo entre o envio de um comando pelo jogador e a efetivação do comando no jogo. A influência deste atraso na percepção do andamento do jogo varia de pessoa para pessoa [PW02b], mas geralmente pode-se estabelecer um valor representativo para o atraso máximo tolerado em um jogo multiusuário. Para jogos representados graficamente como ambientes bidimensionais ou tridimensionais, este atraso é dependente da perspectiva do usuário e da reatividade necessária para conservação da jogabilidade. Baseado nesta perspectiva, dois estilos principais de jogo são definidos: jogos em primeira pessoa e jogos de estratégia em tempo real (também conhecidos por RTS[3]).

Na primeira categoria, o jogador simula a visão do próprio personagem controlado no jogo. Boa parte destes jogos são jogos de tiro, chamados então *First Person Shooter* (FPS), como visto no jogo *Return to Castle Wolfenstein* [Act01]. Esta perspectiva dá ao jogador uma intensa sensação de imersão no jogo, exigindo reflexos rápidos e tempo de respostas quase instantâneo.

[3] Sigla para *Real-Time Strategy*.

Já em jogos de estratégia em tempo real, cada jogador controla um conjunto de unidades independentes, tomando o papel de um comandante ou chefe de uma raça ou exército. O jogador instrui suas unidades (como por exemplo: soldados, veículos, dentre outros) nas ações que cada unidade deve realizar, como atacar inimigos ou deslocar-se para determinadas áreas do mapa. Os jogadores competem entre si para destruir o exército do inimigo ou capturar algum objeto vital no mundo virtual. Nestes jogos, o jogador visualiza o jogo como em uma tomada aérea do mundo virtual e suas ações não necessitam de tempos de resposta tão imediatos quanto jogos FPS. Um exemplo de RTS é o jogo *Alexander* [Ubi04].

Mesmo em jogos de ação que exigem reflexos rápidos, alguns comandos podem possuir requisitos de tempo mais relaxados e podem ser processados com vários segundos de atrasos. Porém, quando se propõem sistemas de suporte para jogos de ação, a preocupação central geralmente é com os comandos que possuem fortes restrições de tempo.

2.2 Cenários atuais para jogos multiusuário

Esta seção apresenta diferentes cenários de utilização de jogos multiusuário. Eles ilustram a evolução tecnológica ocorrida desde o surgimento das primeiras experiências de jogos multiusuário em redes locais até a utilização de dispositivos móveis como plataforma para tais jogos nos dias atuais. Estes cenários levam em conta fatores como o número de jogadores, a utilização de dispositivos móveis e o uso de informações de contexto, especialmente a localização física do jogador.

Esta classificação procura estabelecer uma relação com o tema deste trabalho, pois um jogo multiplataforma multiusuário segue tendências e características pré-estabelecidas nos cenários aqui descritos.

Ressalva-se que, embora experiências com jogos multiusuário por computador existam desde a década de 50 [Wik06], neste levantamento são considerados apenas jogos de tempo real para computadores pessoais e dispositivos móveis, e que apresentem alguma interface gráfica não textual 2D ou 3D, pois estes estão mais diretamente relacionadas com o foco deste trabalho.

2.2.1 Jogos multiusuário em pequena escala

Estes jogos representam a primeira geração de jogos multiusuário para computadores pessoais. Seu surgimento aconteceu em meados da década de noventa, como conseqüência da popularização dos computadores pessoais, do surgimento da multimídia digital e da massificação das redes de computadores. Jogos para PCs até então eram aplicações tipicamente monousuário. Em 1993, porém, a *IdSoftware* [iS06] lançou o jogo *Doom*, que revolucionou o mundo dos jogos para computador, tanto por ser um dos primeiros FPSs lançado no mundo, mas principalmente por introduzir o conceito de jogos multiusuário em redes de computadores.

Nestes jogos, a máquina de um jogador atua como servidor onde a simulação é executada. Esta máquina é responsável por iniciar uma sessão de jogo, a partir da qual jogadores se conectam, permitindo sua interação. Uma característica destes jogos é que cada sessão tem um tempo limitado, determinado por objetivos específicos de cada jogo, como um limite de vinte voltas em uma corrida de carros. Partidas típicas duram alguns minutos ou algumas horas, e o resultado de uma partida não costuma afetar o estado da partida seguinte. Outra característica destes jogos é seu número limitado de jogadores, chegando no máximo a algumas dezenas de usuários em uma única sessão.

Com o surgimento da Internet, estes jogos permitiram que jogadores de diferentes partes do mundo pudessem participar de uma mesma sessão através da rede mundial de computadores. Jogos multiusuário começaram, desta maneira, a criar uma cultura própria. Comunidades foram formadas em torno dos jogos mais populares, como *Quake* [PQ06], *Starcraft* [BE06b] e *Diablo* [BE06a], debatendo estratégias e assuntos correlatos ao jogo. Atualmente, tais jogos representam boa parte dos jogos multiusuário disponíveis para PC, servindo de mola propulsora para o surgimento de estabelecimentos especializados em oferecer suporte adequado em jogos multiusuário, as chamadas *lan-houses*.

2.2.2 Jogos massivamente multiusuário (MMOG)

Jogos massivamente multiusuário (MMOGs[4]) diferem dos jogos multiusuário tradicionais por permitir interação simultânea entre centenas ou milhares de jogadores,

[4] Sigla para *Massively Multiplayer Online Game*.

normalmente em um mundo virtual de estado persistente. A maioria destes jogos é do tipo *Role Playing Game* (RPG), caracterizados por uma narrativa mais complexa, no qual cada jogador assume o papel de um personagem pertencente a uma dentre diferentes categorias, cada uma com características próprias, e deve cumprir missões como encontrar objetos ou destruir inimigos. RPGs focam no comportamento e na evolução de cada personagem, a partir do acúmulo de riquezas e experiências adquiridas durante o cumprimento de suas missões.

O modelo de negócios para MMOGs difere dos jogos tradicionais. Um MMOG é encarado como um serviço, oferecido pela empresa responsável pela infra-estrutura do jogo, cobrado periodicamente de cada usuário pelo seu acesso. Certas vezes, o software cliente é disponibilizado gratuitamente para atrair ainda mais jogadores. Alguns dos mais populares MMOGs, como os jogos da série *Lineage* [NCs06a] [NCs06b] e o *World of Warcraft* [BE06c] ultrapassam 1,5 milhões de usuários [WC03] [Far05].

Porém, apesar de seu sucesso e potencial, MMOGs apresentam uma série de problemas ainda a serem solucionados, principalmente no que diz respeito a questões sobre escalabilidade e custos associados à manutenção da infra-estrutura necessária ao funcionamento do jogo. Estes problemas já fizeram inclusive que grandes empresas cancelassem o desenvolvimento de MMOGs, como a própria Microsoft [Stu04]. Maiores detalhes sobre estes problemas são tratados na seção 3.

2.2.3 Jogos móveis multiusuário

A computação móvel é o cenário mais recente de aplicação de jogos multiusuário. Este cenário é conseqüência de diversos fatores. A popularização de dispositivos móveis, como telefones celulares e computadores de mão, o aumento do poder computacional destes dispositivos e o surgimento de plataformas abertas como Java ME e BREW permitiu a criação de aplicações simples para tais dispositivos, principalmente jogos monousuário, chamados então "jogos móveis". O número crescente de tecnologias de comunicação sem fio como GPRS [GSM06], *Bluetooth* [Blu03] e *Wi-Fi* (802.11) [80105] aproximam cada vez mais dispositivos móveis das redes de computadores tradicionais. Seguindo esta evolução, a inclusão do aspecto multiusuário aparece como conseqüência natural para jogos móveis.

Este processo evolutivo segue os passos dos jogos em computadores pessoais, onde jogos multiusuário surgiram após o amadurecimento e proliferação das redes de computadores tradicionais. O surgimento de plataformas como *Nokia N-Gage* [NG04], *NintendoDS*™

[Nin05] e *Playstation Portable* [SE05] reforça ainda mais esta afirmativa. Estas plataformas são projetadas para o mercado de entretenimento digital, oferecendo diversos jogos multiusuário em escalas que variam de 2 a 16 jogadores, de acordo com a tecnologia utilizada.

De fato, o uso da tecnologia de comunicação acaba sendo um fator determinante no projeto de um jogo multiusuário móvel. Redes de operadoras de telefonia móvel utilizam tecnologias que costumam apresentar latência elevada e alto grau de intermitência na conexão do usuário. Além destes problemas técnicos, existem questões de mercado ainda em aberto, como o modelo de tarifação a ser utilizado entre empresas desenvolvedoras de jogos, distribuidores e operadoras. Por tais motivos, a utilização da rede limita-se a jogos não tão interativos, como jogos em turno (seção 2.1), exemplificado pelo jogo *Batalha Naval* [TdAdAL+03].

Em relação às redes espontâneas (*ad-hoc*) ou redes locais sem fio, *Wireless Local Area Networks* (WLAN), os problemas de latência e intermitência são atenuados, permitindo jogos mais interativos. Contudo, eles oferecem suporte a um número limitado de usuários, devido às restrições das tecnologias utilizadas e abrangência de alcance destas redes.

Apesar das deficiências de cada tecnologia, os resultados da implantação de jogos móveis multiusuário são promissores. Telefones celulares são conectados por natureza, estão sempre perto de seus usuários e podem ser utilizados em praticamente todo lugar e a todo momento. A penetração destes dispositivos na população mundial faz com que estas aplicações possam revolucionar a indústria do entretenimento digital, atingindo uma audiência até então inimaginável, com receitas igualmente proporcionais.

2.2.4 Jogos conectados

Jogos Conectados (*Connected Games*) aparecem como uma alternativa para criação de jogos multiusuário em dispositivos móveis mesmo em face aos problemas de conexão de redes sem fio. Nestes jogos, a jogabilidade básica não requer uma conexão permanente com outras entidades, como servidores ou outros jogadores, mas funcionalidades suplementares são acessadas através de uma rede. Exemplos de funções possíveis são a publicação da pontuação dos jogadores ao final de cada partida ou o *download* de novos níveis de dificuldade ou itens do jogo.

Neste caso, não há interação direta entre os jogadores, o que de certa forma acaba comprometendo o uso do termo “multiusuário” para tal cenário. Porém, a idéia de jogos

conectados permite que jogadores possam competir indiretamente, como na publicação das maiores pontuações ou na ativação de níveis mais avançados.

Um exemplo de um jogo conectado desenvolvido no Brasil é o "Meu Big Brother" [MG05], onde cada jogador escolhe um personagem associado a um dos participantes do conhecido *reality show* de televisão para controlar. Cada jogador deve manter seu personagem limpo, bem-humorado, dando presentes ou punindo-o quando ele não estiver se comportando. O jogo interage em tempo real com o programa, podendo ser influenciado pelos eventos do mesmo. O jogo permite que cada jogador receba informações sobre o programa, notícias, questionários e enquetes. O desempenho de cada jogador é enviado para um servidor que estabelece o ranking entre os jogadores.

2.2.5 Jogos móveis massivamente multiusuário (3MOG)

Jogos móveis massivamente multiusuário, 3MOGs são jogos que utilizam as redes das operadoras de telefonia celular de modo a permitir o suporte a um alto número de jogadores. De certa forma, as operadoras possuem um interesse especial nos jogos multiusuário, pelo potencial de geração de tráfego adicional de dados nas suas redes e conseqüente aumento da receita.

Os problemas decorrentes das infra-estruturas de redes de telefonia podem ser comuns. No entanto, são atenuados através de projetos adequados às limitações existentes, como o uso de batalhas por turnos ou retardo na execução das ações dos jogadores.

Um dos primeiros 3MOGs comerciais foi *Samurai Romanesque* [Kri03], lançado no Japão em 2001. Destaca-se por incorporar uma série de soluções para os problemas de limitação técnica dos celulares, como a utilização de vários módulos separados que representam cada fase do ciclo de vida do jogador. Outro 3MOG de destaque é *Tibia Micro Edition* [Nok03], concebido como a versão móvel de um conhecido jogo para PCs. Permite situações de interação entre jogadores, mesmo com alta latência, através do uso de esquemas de estimativas. Há também o *Pocket Kingdom* [Nok06a], um 3MOG desenvolvido pela *Sega.com*, divisão independente da *Sega Inc.* [Seg06]. Após sua aquisição em 2003 pela Nokia com o objetivo de agregar as competências de jogos da empresa e alavancar sua presença na produção de jogos para telefones celulares, o jogo foi projetado apenas para a plataforma *Nokia N-Gage*.

2.2.6 Jogos pervasivos

A contínua evolução tecnológica que transforma o cotidiano da sociedade moderna impulsiona novos tipos de jogos eletrônicos. Um conceito mais recente neste sentido é o de jogos pervasivos ou jogos ubíquos. Estes jogos aproveitam a visão de Computação Pervasiva [IBM99] para criar novas formas de interação e entretenimento que vão além daquelas realizadas em frente a computadores ou consoles de *videogame*, permitindo que jogadores desloquem-se e interajam de diferentes formas, através de diversos dispositivos e tecnologias de comunicação. Seu campo de aplicação é um ambiente real, como uma sala, um prédio ou uma cidade, de forma semelhante a antigos jogos tradicionais de crianças (perseguição, captura de bandeira etc), mas com acréscimo de serviços computacionais em tempo real.

Estas aplicações combinam o real e o virtual em jogos que aproveitam os inovadores aparatos tecnológicos para utilizar informações do mundo real como elemento influente no mundo virtual que compõe o jogo. Os jogos são concebidos com base em três tecnologias principais: dispositivos móveis, comunicação sem fio e tecnologias capazes de capturar informações no contexto do mundo real dos jogadores, particularmente sua localização física.

Muitos dos experimentos relacionados com jogos pervasivos trazem conceitos de realidade virtual, como utilização de dispositivos como capacetes ou utensílios que permitem mesclar objetos do mundo virtual na visão dos jogadores. Em exemplo é o *Human Pacman* (“Come-Come Humano”) [CFG+03], que apresenta uma versão do conhecido jogo *Pacman*, mesclando o mundo real com o mundo virtual. Os personagens do jogo são jogadores que se movem em uma área do mundo real, mapeada em um mapa virtual. O labirinto e os elementos do jogo são embutidos no mundo real, criando uma experiência inovadora aos jogadores. A informação do mapa é distribuída através de uma WLAN por um servidor central. Os movimentos e as posições de cada jogador são obtidos através de GPS [GPS06] e enviados ao servidor que mantém o estado do jogo. Este estado é repassado a todos os jogadores. Cada jogador usa um visor que apresenta o mundo real mesclado com elementos do mundo virtual.

2.2.7 Jogos baseados em localização

Dentre as tecnologias que aparecem como mais promissoras como veículos de jogos pervasivos estão os telefones celulares, visto que estes são, sem dúvida, a plataforma mais ubíqua da sociedade moderna. O lançamento de serviços de localização (LBS, ou *Location*

Based Services) pelas operadoras de telefonia celular gerou uma busca por aplicações que utilizassem seus recursos e que chamassem a atenção do mercado consumidor. Assim, foi possível o surgimento de jogos com características pervasivas via telefones móveis.

Típicos jogos LBS oferecem suporte a um número alto de jogadores interagindo com seus telefones celulares, em batalhas realizadas nas ruas e bairros de uma cidade. As oportunidades abertas por estes jogos são vastas, em termos de novas oportunidades de lazer, novos modelos de negócio e suporte a comunicação em grupo para aprendizado, socialização e atividades culturais. Jogos LBS representam um emergente mercado comercial para jogos.

As primeiras experiências destes jogos ocorreram nos mercados europeu e asiático, onde a cultura de jogos é mais forte. O primeiro jogo deste tipo foi *Botfighters* [Day06], lançado em 2000 e desenvolvido pela empresa sueca *It's Alive*, que foi posteriormente incorporada por outra empresa sueca chamada *DayDream*. O jogo simula batalhas entre robôs nas ruas de uma cidade, e permite o uso de celulares tanto com tecnologia SMS [SMS99] como Java. Outro exemplo semelhante e de grande sucesso foi *Undercover* [YDr06b], lançado em junho de 2003 pela empresa portuguesa *YDreams* [YDr06a], que teve cerca de 5000 inscritos nos 3 primeiros dias após o seu lançamento. No Brasil, o primeiro experimento de um jogo massivo móvel foi *Alien Revolt* [MC05]. Através da tecnologia de localização, o jogo possibilita a localização de jogadores através de um mapeamento de uma região, como o estado do Rio de Janeiro.

2.3 Aspectos técnicos de jogos multiusuário

Sendo aplicações de espaço compartilhado, é fundamental para jogos multiusuário manter o estado global consistente entre os jogadores. Para isto, as ações de cada jogador devem ser repassadas entre os jogadores de forma rápida, confiável e ordenada, de modo a garantir a correta execução dos eventos no jogo. Porém, a separação física dos componentes e jogadores decorrente da natureza distribuída de ambientes virtuais em rede impõe obstáculos para a manutenção de um estado global consistente entre os jogadores.

Desta forma, diversos aspectos como latência, largura de banda, escalabilidade e segurança têm um impacto significativo sobre as técnicas existentes para implementação de jogos multiusuário. A seguir são detalhados como tais aspectos influenciam a implementação

de jogos multiusuário em geral, e são também apresentadas as técnicas mais utilizadas para atenuar o impacto destes problemas.

2.3.1 Fatores limitantes para jogos multiusuário

Helin [Hel03] aponta três grandes problemas a serem enfrentados por jogos multiusuário em rede. Primeiramente, a infraestrutura de rede afeta diretamente o andamento e consistência do jogo. Em seguida, a arquitetura de comunicação entre jogadores, servidores e demais componentes criam obstáculos para escalabilidade em jogos multiusuário. Por fim, a necessidade de garantia de um jogo justo através da prevenção de trapaça por certos jogadores constitui um problema relevante para jogo multiusuário. Cada um destes problemas é melhor abordado a seguir:

- **Plataforma Física**

A plataforma física de suporte ao jogo trata-se da infraestrutura de rede utilizada. Esta plataforma apresenta limites com os quais jogos multiusuário devem conviver, como latência e largura de banda.

Largura de banda é definida como a capacidade de transmissão através de uma linha de comunicação. Para jogos multiusuário esta medida é de fundamental importância, uma vez que representa um limitador na quantidade de informação que pode ser transferida entre os jogadores. Em redes locais, a largura de banda atinge altas taxas de transmissão, diferentemente da Internet. Além disso, a largura de banda necessária para um jogo multiusuário depende também do número de jogadores e das técnicas de distribuição de mensagens utilizadas no jogo (seção 2.3.2).

Por sua vez, latência caracteriza-se pela medida de tempo entre o envio da mensagem e sua recepção pelo destinatário. Este atraso decorrente da inerente separação física entre os participantes existe em várias escalas e nunca poderá ser totalmente eliminado. Para jogos multiusuário, a latência na troca de informações afeta diretamente a percepção de andamento e continuidade do jogo, influenciando negativamente o nível de interatividade (ou reatividade) ao qual os jogadores estarão sujeitos. De fato, avanços em dispositivos multimídia como placas de vídeo e som mais apurados, bem como monitores de alta resolução, permitiram que a sensação de imersão nos jogos eletrônicos aumentasse cada vez mais. Porém, para jogos multiusuário, mesmo possuindo o melhor hardware disponível no mercado, a sensação de

imersão será totalmente comprometida caso o andamento do jogo seja constantemente interrompido por atrasos na troca de informações entre os jogadores.

Para aplicações interativas distribuídas, pesquisas mostram que atrasos nas trocas de dados não devem ultrapassar 200 ms [SKH01], sob pena de afetar a percepção de continuidade para o usuário. Porém, para jogos multiusuário este número varia de acordo com o estilo do jogo. Em jogos de estratégia em tempo real, latências de até 350 ms são aceitáveis, contanto que se mantenham estáveis. Entretanto, em jogos de alta interação e de maior imersão como aqueles em primeira pessoa, a latência média deve ficar próxima a 100 ms [SKH01] [PW02a]. Para jogos em turno, o problema da reatividade é mais simples, uma vez que as ações de cada jogador acontecem em intervalos de tempo controlados e previsíveis. Porém, a consistência do estado do jogo é imprescindível. Para jogos em tempo real, a consistência do estado global pode ser relaxada em benefício da reatividade de cada jogador. Como exemplo, jogadores podem utilizar estimativas (previsões) sobre a posição do avatar de um jogador para permitir um fluxo contínuo no andamento do jogo.

- **Plataforma Lógica**

Além das limitações impostas pela infraestrutura de rede, as arquiteturas de comunicação (cliente/servidor, ponto a ponto ou clusters de servidores), bem como arquiteturas de controle e de dados (centralizado, replicado ou distribuído) formam a plataforma lógica, criando outros fatores limitantes a serem considerados, principalmente em relação à escalabilidade de um jogo. Enquanto não há muito a ser feito em relação à plataforma física (a não ser investir em hardware e/ou rede, quando de sua propriedade), a plataforma lógica assume um papel fundamental para atenuar os efeitos prejudiciais que a plataforma física pode impor, como é visto na seção 2.3.2.

- **Segurança e Trapaça**

Questões sobre a segurança em jogos multiusuário vêm se tornando uma das maiores preocupações para os envolvidos no desenvolvimento e manutenção de tais aplicações. Enquanto para jogos monousuário as maiores preocupações recaem sobre pirataria, para jogos multiusuário novos problemas surgem. Primeiramente, jogos *online* estão sujeitos a problemas como a segurança dos dados confidenciais de seus usuários, autenticidade e disponibilidade dos sistemas que hospedam o próprio jogo. Estes problemas são comuns a outros domínios de aplicação como comércio eletrônico ou agências bancárias na Internet, onde medidas já utilizadas, como criptografia, também se aplicam a jogos multiusuário.

Porém, um segundo problema de segurança é peculiar apenas a jogos *online*. Relaciona-se à possibilidade de jogadores trapaceiros utilizarem-se de recursos ou meios antiéticos no intuito de obter vantagens ilegais sobre os demais jogadores. Embora o conceito de trapaça já exista e seja tolerável em jogos monousuário, seu efeito para jogos multiusuário é completamente indesejável. Jogos monousuário permitem que jogadores utilizem códigos e ferramentas para facilitar a realização de certos objetivos sem maiores esforços. Para jogos multiusuário, jogadores trapaceiros criam desequilíbrio, desencorajando os demais jogadores, principalmente iniciantes, a participarem do jogo. Num mercado onde o jogador deve ser estimulado a permanecer o maior tempo possível no jogo, a facilidade de trapacear pode arruinar a viabilidade comercial de um jogo [Yan03].

A trapaça em um jogo multiusuário pode ocorrer de várias formas. Pritchard [Pri00] criou uma primeira classificação para as formas de trapaça existentes em jogos multiusuário. Yan & Choi [YC02] estenderam esta classificação citando 11 diferentes categorias. Certas categorias envolvem aspectos sociais e não se relacionam com alterações no software ou falhas em sua concepção. Para estes tipos de trapaça, é difícil imaginar medidas preventivas, como diagnosticar trapaça por conivência em que, por exemplo, um administrador de um *site* de jogos vende clandestinamente senhas de acesso. Porém, certos mecanismos podem ser utilizados para dificultar as ações de *hackers*. Mecanismos tradicionais de segurança como criptografia, assinaturas digitais e controle de integridade podem ser bem empregados para jogos *online*, embora individualmente não possam ser considerados como soluções definitivas.

Yan & Choi [YC02] apresentam o termo de "Mitigação de Trapaça", através do qual é proposta uma abordagem sistemática para prevenção, detecção e gerenciamento de trapaça em jogos *online*. Nesta abordagem, ressaltam-se medidas simples – como o uso de políticas para definições de senhas dos jogadores – e algumas mais complexas e controversas, como o uso de análises estatísticas para detecção de jogadores que sejam bons demais para não estarem trapaceando. Entretanto, estas soluções para problemas de trapaça não serão detalhadas, pois não fazem parte do escopo deste trabalho.

2.3.2 Principais soluções para suporte a jogos multiusuário

De acordo com Smed [SKH01], os conceitos relativos à plataforma lógica impõem limites que devem ser tratados em níveis mais altos que aqueles que lidam com a plataforma

física de um jogo multiusuário. O uso de diferentes topologias e técnicas de distribuição de mensagens permite a definição de arquiteturas de comunicação, de controle e dados, que ajudar a atenuar os efeitos destes limitadores.

- **Distribuição de mensagens**

A largura de banda necessária para um jogo multiusuário depende do número de jogadores e das técnicas de distribuição de mensagens utilizadas no jogo. A figura 1 apresenta as principais formas de distribuição das mensagens entre os jogadores de um jogo multiusuário.

A primeira técnica, conhecida como *broadcast* (figura 1 (a)), repassa a mensagem de um jogador para todos jogadores presentes na partida. Ao receber uma mensagem, cabe ao jogador utilizá-la ou descartá-la de acordo com seu conteúdo. Esta técnica gera uma má utilização da capacidade de transmissão da rede, visto que mensagens são enviadas para todos jogadores e podem ser irrelevantes para a maioria. Este problema acentua-se no caso do aumento de jogadores, criando um sério problema de escalabilidade e desempenho, devido ao congestionamento da rede para jogos que utilizem tal abordagem.

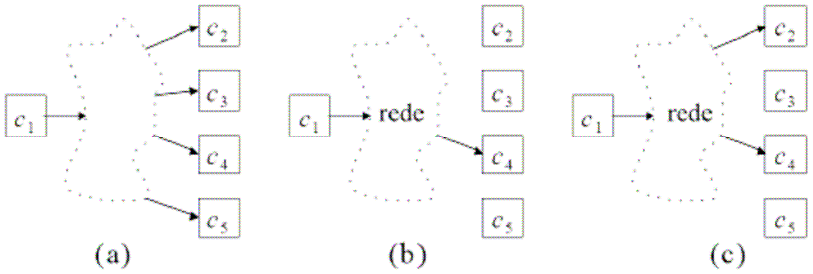

Figura 1 - Técnicas para transmissão de mensagens. (a) *Broadcast*, (b) *Unicast* e (c) *Multicast*

Uma segunda técnica, chamada *unicast*, representada pela figura 1 (b), determina que cada mensagem possua um único emissor e um único receptor. Embora mais eficiente que broadcast no envio de mensagens para um único destinatário, torna-se desinteressante quando há múltiplos destinatários para uma única mensagem, caso comum na maioria dos jogos multiusuário. Neste cenário, *unicast* utiliza mensagens redundantes, fazendo com que novamente haja uma sobrecarga na rede.

A última técnica, chamada *multicast* e apresentada na figura 1 (c), permite que uma única mensagem seja distribuída para um grupo de destinatários, permitindo uma melhor

utilização da rede. Apesar de parecer a melhor solução para a comunicação entre os envolvidos no jogo, *multicast* apresenta como problemas a falta de suporte pela atual infraestrutura da Internet para sua utilização, bem como questões de segurança relacionadas com implementação de *multicast* sobre uma rede TCP/IP.

- **Arquiteturas de Comunicação**

A arquitetura de comunicação de um jogo multiusuário baseia-se nas diferentes formas como os computadores envolvidos em um jogo estão interconectados. A figura 2 ilustra as mais conhecidas arquiteturas para jogos multiusuário [Hel03].

A figura 2 (a) apresenta a arquitetura ponto a ponto. Neste modelo, a topologia de comunicação entre jogadores é formada por um conjunto de nós, onde todos possuem características iguais em relação ao software necessário para participação no jogo. Logo, cada nó precisa se comunicar com todos demais participantes para trocar informações. Como exemplo, para enviar uma mensagem aos demais jogadores, um jogador realiza um broadcast (ou *multicast*) de sua mensagem para cada participante do jogo. Cada participante possui uma cópia local do estado do jogo, fazendo com que seja essencial o uso de esquemas de sincronização para garantir a consistência do estado do jogo entre os jogadores. Devido a tal característica, arquiteturas ponto a ponto apresentam certos problemas de escalabilidade: primeiro, o andamento do jogo é determinado pelo nó com pior taxa de transmissão entre os usuários. No caso da Internet, é comum ter entre os jogadores, usuários com baixas taxas de transmissão, prejudicando o andamento do jogo como um todo. Um segundo fator é que não se espera que as máquinas dos jogadores tenham capacidade para manipular adequadamente um número muito alto de conexões, como por exemplo no caso de um jogo com centenas ou milhares de usuários [FRC03]. Mesmo com tais limitações, esta abordagem foi utilizada por jogos de estratégia em tempo real, como *Age of Empires* [Mic04], onde o número de jogadores é limitado e o tempo de resposta não é tão restrito.

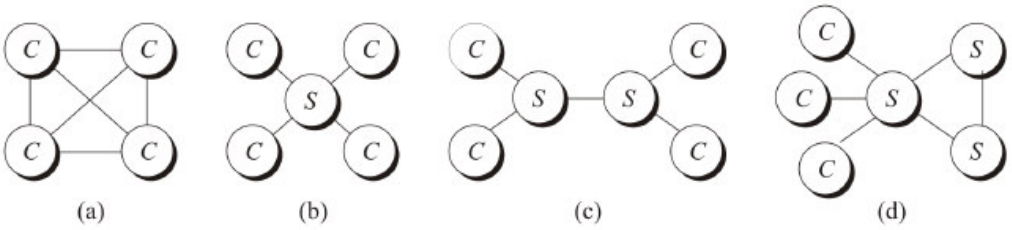

Figura 2 - Arquiteturas de comunicação para jogos. (a) Ponto a Ponto, (b) Cliente/Servidor, (c) Replicação de servidores e (d) *Grid* de servidores.

Na arquitetura cliente/servidor, representada pela figura 2 (b), um nó da rede é promovido ao papel de servidor do jogo, responsável pela comunicação entre os jogadores. O servidor mantém o estado do jogo centralizado e recebe notificações sobre as atualizações de cada jogador. A partir destas informações, o servidor atualiza o estado do jogo e repassa as atualizações para os demais jogadores. Apesar de possuir implementações diferentes para os jogadores e o servidor, tal modelo traz certas vantagens em relação ao modelo ponto a ponto: Primeiro, funcionalidades administrativas desejáveis, como controle de acesso ao jogo e tarifação de jogadores, são mais facilmente implementadas em uma topologia cliente/servidor. Outra vantagem é que a ação de um jogador só precisa ser comunicada ao servidor, fazendo com que seu tempo de resposta seja consideravelmente diminuído. Por estas características, jogos em primeira pessoa são tradicionalmente implementados usando a arquitetura cliente/servidor. Por fim, a manutenção do estado do jogo no servidor torna a arquitetura bem resistente a ações de jogadores trapaceiros.

No entanto, o modelo cliente/servidor também apresenta problemas: o primeiro e o mais observável, é a criação de um potencial risco representado pela possibilidade de falha ou indisponibilidade do servidor. Na arquitetura ponto a ponto, como não há centralização do estado do jogo, a queda de um nó não representa risco à continuidade do jogo. Um segundo problema é a latência adicional representada pelo processamento do estado do jogo pelo servidor, caso seu poder computacional não seja suficiente para tratar as requisições dos jogadores.

Dependendo da audiência, o custo do servidor será um problema. Jogos que utilizam servidores dedicados para suporte a milhares de usuários *online* apresentam custos anuais de manutenção de até US$ 10 milhões em hardware, bem como alta demanda de recursos de rede para suportar tal escala [IDG02]. Apesar destes fatores, arquiteturas cliente/servidor são utilizadas na maioria dos jogos comerciais atuais ou então através de clusters de servidores, como veremos a seguir.

A figura 2 (c) ilustra a arquitetura de replicação de servidores, que representa um modelo híbrido entre as arquiteturas anteriores. Nesta, a infraestrutura de comunicação dos jogadores pode ser vista como uma arquitetura ponto a ponto de servidores de arquiteturas cliente/servidor. Com isto, acaba-se com o potencial risco de se possuir um único ponto de falha como na arquitetura cliente/servidor convencional. No caso da queda de um servidor, jogadores podem ser realocados para outro servidor. Este mesmo princípio é válido para o

balanceamento de carga entre servidores sobrecarregados ou subutilizados. A arquitetura de replicação de servidores reduz os requisitos computacionais impostos ao servidor, promovendo uma maior escalabilidade do jogo. Porém, como efeito colateral, a complexidade para a gerência do tráfego das informações entre jogadores e servidores é consideravelmente aumentada, além de na maioria dos casos, não permitir uma comunicação direta de jogadores em diferentes servidores.

Por fim, a figura 2 (d) apresenta o modelo onde os servidores do jogo são organizados de modo a formar um *grid* computacional onde o estado do jogo é distribuído entre os vários servidores. De acordo com a alocação dos jogadores, mais servidores podem ser alocados de forma a dar suporte adequado à entrada de novos jogadores. Da mesma forma, se houver uma saída de jogadores, os servidores podem ser realocados para outras aplicações ou outros jogos dinamicamente. O uso de *grid* computacional põe fim a um dilema onde a empresa hospedeira de um jogo acabe com poder computacional de sobra no caso de um jogo com poucos usuários, ou com uma sobrecarga de servidores quando do sucesso do jogo. Outra vantagem é que o uso de computação em grade elimina as fronteiras entre jogadores, deixando transparente aos desenvolvedores e jogadores as partições do mundo virtual. Porém, a utilização de *grids* não acaba com o alto consumo de banda do lado servidor do jogo.

- **Compressão e Agregação de Mensagens**

Enquanto a compressão das mensagens diminui o tamanho das mensagens transmitidas através da redução do número de bits necessários para representar a informação enviada, a agregação mescla informações de várias mensagens em uma única mensagem maior, porém diminuindo o *overhead* gerado com cabeçalhos de mensagens. Dependendo do tamanho do cabeçalho e do número de mensagens mescladas, uma economia considerável na largura de banda pode ser alcançada. As duas técnicas procuram diminuir o consumo da banda utilizada ao custo de processamentos e atrasos adicionais pela execução de tais procedimentos.

- **Gerenciamento de Áreas de Interesse**

Dependendo do projeto do jogo, o mundo virtual pode ser dividido em áreas menores. Nesta subdivisão, as áreas de interesse para um jogador são aquelas próximas a sua localização dentro do mundo virtual. O conceito de área (ou aura) de interesse refere-se a parte do mundo com a qual o usuário pode interagir, e conseqüentemente, que é de seu interesse. O gerenciamento destas áreas permite que os jogadores expressem seus interesses em subconjuntos de informações do jogo.

Sem esta divisão, o servidor teria que repassar todos os eventos ocorridos no mundo virtual, ocasionando um alto consumo de largura de banda e tempo de CPU. O esquema de gerenciamento e filtragem de mensagens pode ser intrínseco (no nível da aplicação) ou extrínseco (no nível dos protocolos de rede). No primeiro, o filtro utiliza informações específicas da aplicação para determinar quais nós devem receber uma mensagem, proporcionado informações bem apuradas a um custo maior de processamento. Neste esquema, o gerenciamento é tipicamente feito através de uma divisão do mapa do jogo em regiões menores, onde um jogador só recebe e repassa informações aos jogadores que estejam na sua mesma região. No segundo caso, os receptores de uma mensagem são determinados pelos seus atributos de rede, como seu endereço IP. *Multicasting* (figura 2 (c)) é um exemplo de tal esquema.

- **Dead Reckoning**

Outra forma de atenuar o consumo dos recursos da rede é diminuir a freqüência de envio de mensagens entre jogadores. Embora a princípio esta abordagem traga problemas em relação à consistência do jogo, é possível utilizar técnicas de previsão para diminuir o impacto da ausência de informações de atualização. Para isto, *dead reckoning*, técnica navegacional que utiliza estimativas de cálculo para o estado atual de um objeto a partir de dados anteriores, podem ser utilizadas para diminuir o tráfego de mensagens e melhorar a reatividade para os jogadores. Neste caso, a consistência é de certa forma relaxada em benefício de um melhor andamento do jogo. Dependendo da qualidade dos algoritmos de previsão utilizados, objetos podem ser postos em posições diferentes da sua real localização. Para correção destes erros é estabelecida uma tolerância na diferença entre os valores previstos e os valores reais. Enquanto a diferença se mantiver dentro do valor tolerado, o usuário não envia informações para os demais jogadores. Caso contrário, as devidas correções são feitas através de algoritmos de convergência, de modo a eliminar possíveis inconsistências e tornar imperceptível o erro ao jogador.

Os esquemas de previsão podem basear-se tanto na entrada do jogador quanto nas últimas informações recebidas do objeto controlado, como direção, velocidade e aceleração. Para cada jogador, previsões são realizadas em sua máquina local, de acordo com algoritmos de simulação e entradas realizadas pelo usuário. Pantel e Wolf [PW02b] apresentam um estudo que mostra que fatores como o valor da tolerância, bem como o estilo do jogo são bastante influentes para o sucesso de *dead reckoning* para jogos multiusuário.

2.4 *Middlewares* para jogos multiusuário

As dificuldades para o desenvolvimento de jogos multiusuário demandam soluções que aliviem o trabalho dos desenvolvedores de tais aplicações. Neste sentido, cada vez mais tem surgido *middlewares* que se propõem a facilitar o desenvolvimento de jogos multiusuário.

Existem várias definições para *middleware* [Ber96] [Emm00] [CDK05b], que convergem para a idéia de uma camada de software que facilite o desenvolvimento de aplicações distribuídas, abstraindo do desenvolvedor problemas relativos à distribuição de seus componentes. Esta é, pois, a definição utilizada neste trabalho.

Embora apareçam atualmente principalmente como produtos comerciais, já são encontrados esforços para criação de *middlewares* para jogos multiusuário de código aberto. Alguns destes são descritos a seguir:

- **DirectPlay**

DirectPlay é parte integrante do DirectX [Dir06], uma solução da Microsoft para construção de aplicações multimídia sobre a plataforma Windows e Xbox. O DirectPlay é responsável por abstrações para desenvolvimento de aplicações multiusuário, incluindo jogos. Este componente provê um modelo de comunicação independente de rede e dispositivo através de uma API que disponibiliza serviços no nível das camadas de transporte e sessão. A comunicação utiliza o conceito de sessão, que representa uma instância de um jogo multiusuário em uma rede. Esta sessão estabelece o canal de comunicação entre vários usuários de um mesmo jogo. DirectPlay provê uma série de funções que simplificam a implementação de vários aspectos de uma aplicação multiusuário, como:

  - Criação e gerenciamento de sessões utilizando topologias cliente/servidor ou ponto a ponto;
  - Gerenciamento de usuários individuais ou em grupos dentro de uma sessão de comunicação;
  - Gerenciamento da troca de mensagens entre membros de uma sessão, mesmo através de diferentes tipos de rede e sobre diferentes condições de tráfego;
  - Uso de comunicação via voz;

- **TeraZona**

TeraZona [Ter02] é a solução desenvolvida pela empresa Zona Inc., para o suporte a MMOGs. Através de um arcabouço de software, é fornecida uma camada abstrata de rede

especialmente projetada para resolver questões de escalabilidade e confiabilidade de MMOGs. Segundo dados da empresa, a solução TeraZona permite até 32 mil jogadores por aglomerado de servidores, podendo suportar através da união deste aglomerados centenas de milhares de jogadores.

A arquitetura TeraZona é constituída de múltiplas camadas, cada uma com uma responsabilidade específica. Cada camada é formada por componentes extensíveis e reutilizáveis baseados em objetos distribuídos, agrupados por funcionalidades, oferecendo seus serviços via uma interface. Segurança é conseguida por criptografia de mensagens, e certos tipos de trapaça podem ser combatidos pelo monitoramento do comportamento do jogador.

O *framework* TeraZona é dividido em duas partes: um relativo ao processo servidor do jogo e outro relativo aos jogadores. A API do lado servidor trata de questões necessárias para a efetiva distribuição do estado do jogo entre os jogadores. O lado servidor do *framework* recebe os estados de cada jogador, e utiliza seus componentes para integrá-los ao jogo.

- **GASP**

GASP (*GAming Services Platform*) [PDGSS05] é um *middleware* desenvolvido pelo grupo ObjectWeb [Obj05], que permite a implementação de jogos multiusuário para dispositivos móveis baseados na topologia cliente/servidor, que utilizam a rede GPRS de uma operadora de telefonia, como meio de comunicação do jogo. GASP é um projeto de software aberto baseado na especificação dos Serviços de Jogos do consórcio *Open Mobile Alliance* (OMA) [OMAOb] em conjunto com o *Mobile Games Interoperability Forum* (MGIF) [OMAOa].

O consórcio OMA foi formado em 2002 por uma associação de mais de 200 empresas de diversas áreas de atuação: operadores de telefonia móvel, empresas de manufatura de dispositivos móveis, empresas especializadas em redes de comunicação, nas tecnologias da informação e fornecedores de conteúdo, como Nokia, IBM, Motorola, SIEMENS, Intel e Sun. Este consórcio visa realizar esforços de padronização motivados pela vontade de homogeneização e interoperabilidade de tecnologias para computação móvel.

Por sua vez, MGIF, grupo já existente formado por Ericsson, Motorola, Nokia e SIEMENS, tinha o objetivo de criar serviços específicos para os jogos móveis. Este grupo juntou-se ao consórcio OMA, criando o grupo de trabalho de Serviços de Jogos - Games Services. O grupo lançou uma especificação de uma arquitetura de jogos para dispositivos

móveis baseada na topologia cliente/servidor, com dois objetivos prioritários: a interoperabilidade, a fim de compartilhar recursos ou serviços, e a mobilidade. Este arquitetura OMA/MGIF é especificada na forma de uma API Java, com interfaces para um conjunto de serviços básicos. Dentre alguns serviços, pode-se citar: gerência de sessão, conectividade, tarifação, *logging*, temporização e gerência de pontuação. O GASP é uma implementação parcial desta arquitetura proposta.

Na proposta GASP, as interfaces de comunicação da especificação OMA relativos à lógica do jogo entre cliente e servidor são respeitadas. Porém, o modelo de funcionamento interno da arquitetura é estendido. Este modelo permite um esclarecimento dos papéis de cada classe e uma otimização do número de classes instanciadas na arquitetura.

CAPÍTULO 3

Jogos multiplataforma multiusuário

Atualmente, jogos multiplataforma multiusuário já são uma realidade, já tendo sido realizadas inclusive várias iniciativas comerciais. Introduzindo o aspecto multiplataforma, estas iniciativas permitem o acesso de jogadores a partir de dispositivos distintos, como *desktops* e aparelhos celulares. Contudo, justamente por estarem desbravando um novo caminho, não exploram todo o potencial desta área.

Alguns trabalhos já foram propostos a respeito, visando explorar tudo o que o aspecto multiplataforma pode oferecer, mas trata-se ainda de uma área incipiente. Não há uma especificação dos serviços, sequer cenários de utilização já detalhados.

Nesta seção são citados alguns exemplos destes jogos multiplataforma multiusuário já comercializados e também alguns trabalhos relacionados ao tema. Em seguida, é introduzida e descrita a proposta PM³G.

3.1 Exemplos comerciais de jogos multiplataforma multiusuário

Lineage Mobile é uma versão para celulares do *Lineage* [NCs06b], um dos maiores sucessos mundiais no segmento MMOG. Foi projetado especificamente para celulares de última geração, demandando grandes recursos de processamento nos aparelhos e redes de transmissão de alta velocidade. Baseado na temática de masmorras e dragões comum em jogos RPG, o jogo permite que os pontos feitos pelos participantes nas partidas por celulares sejam também contabilizados na versão para computadores tradicionais.

Também para o MMOG *Ragnarok Online* [GC06] foi criada uma versão para aparelhos celulares, o *Ragnarok Mobile Mage*. Neste jogo, ao contrário do que se poderia imaginar, não se controla o mesmo personagem do *Ragnarok Online*; ao invés disso, deve-se criar um

personagem específico para ser controlado pelo aparelho celular. O mundo do jogo é um só, e uma forma de fazer os personagens de *Ragnarok Mobile Mage* e *Ragnarok Online* interagirem entre si é a possibilidade de se transferir dinheiro de um para o outro.

Esta foi a forma encontrada por empresas criadoras de jogos famosos de estender a atuação deles e ampliar o número de clientes. Desenvolver e disponibilizar uma versão de um MMOG para celulares pode parecer uma idéia interessante, pois são mantidos os usuários, o mundo virtual e as regras básicas do jogo original. Contudo, tal abordagem pode se tornar complexa, havendo a necessidade de uma adaptação do projeto do jogo para incluir uma nova gama de jogadores que participam do jogo em um cenário diferente dos demais. Conseqüentemente, mudanças na interação, na apresentação e na freqüência de uso do jogo devem ser levadas em consideração, permitindo uma interação justa entre jogadores que utilizem diferentes dispositivos.

Jogos originalmente concebidos como multiplataforma também já se encontram no mercado. São os casos de *Mogi, Item Hunt* [NG06] e *HinterWars* [Nok06b]. O primeiro trata-se de um jogo que utiliza serviços de localização para criar um ambiente no qual jogadores são incentivados a colecionar itens virtuais espalhados em diversos pontos da cidade de Tóquio. Os itens são agrupados em coleções e variam em valor de acordo com a raridade do mesmo no mundo virtual. O jogo possibilita ainda que os jogadores troquem itens para completar coleções. A característica multiplataforma está no fato que jogadores por meio de computadores pessoais podem utilizar o *site* do jogo para monitorar a posição de jogadores móveis e ainda auxiliá-los, indicando a presença de itens nas cercanias do jogador monitorado. Deste modo, o acesso principal ao jogo é por dispositivos móveis, sendo o acesso pela *web* não uma característica essencial, mas apenas adicional.

HinterWars é um jogo massivo realizado em uma parceria entre a Nokia e a empresa de desenvolvimento *Activate Interative*. Permite que milhares de jogadores participem do jogo, tanto através de seus PCs como de seus celulares *N-Gage*™. O jogo permite também bate-papo confidencial e negociação de itens virtuais do jogo. De acordo com a evolução do jogador, bonificações são incluídas ao seu perfil, como artigos raros, poderes adicionais e experiência, recompensas virtuais e fama dentro da comunidade do jogo. Produzido com o objetivo de promover a plataforma *N-Gage* na Ásia, a Nokia desenvolveu para ela características adicionais, como disponibilidade de equipamentos especiais para o personagem e um aumento mais rápido de seus atributos.

Os exemplos acima demonstram o quão recente é a questão de multiplataforma em jogos. Enquanto alguns não foram originalmente projetados com esta característica, outros focam em um dispositivo específico. Falta a todos eles, portanto, uma visão mais ampla de pervasividade incorporada a jogos, embora já seja discutida em teoria, como é mostrado na seção a seguir.

3.2 Trabalhos de pesquisa relacionados

A incorporação de mobilidade ou pervasividade em jogos multiusuário tradicionais ainda é muito recente. Poucos são os trabalhos que apresentam estudos, propostas ou experimentos sobre o tema. Muitos dos trabalhos publicados na literatura fazem um levantamento das vantagens a serem alcançadas com tal integração, porém sem soluções para sua concretização [Fox03] [Wal04].

Fox [Fox03] foi o primeiro a propor a integração entre MMOGs e dispositivos móveis. Dentre os pontos destacados em seu artigo, o autor ressalta a necessidade da adaptação da jogabilidade de jogadores via diferentes dispositivos em um MMOG. A jogabilidade englobaria não apenas a forma como o jogador visualiza o jogo, mas também como cada jogador interage com o mundo virtual. No entanto, este trabalho apresenta suas soluções em um alto-nível, sem o detalhamento de como suas visões seriam de fato realizadas.

Em seguida, Han et al [HIW04] apresentam uma primeira proposta de um *middleware* sensível a situação ("*situation-aware middleware*") para jogos multiplataforma. Cinco tipos de plataforma são suportadas: consoles, máquinas comerciais do tipo arcade, computadores pessoais, PDAs e telefones celulares. Os jogadores de uma plataforma podem migrar o estado de seus jogos entre diferentes plataformas, de acordo com a situação do jogador. A arquitetura proposta por Han et al define a existência de um servidor que responde pela lógica do jogo, bem como por serviços de tarifação e persistência. O *middleware* utiliza informações de sensores que detectam situações (como presença de um dispositivo melhor adequado para o jogador), e comportamento associado a cada situação (por exemplo, transferência do estado entre dispositivos). Porém, o conceito de situação e a falta de dados sobre experimentos realizados pela pesquisa não permite que seja feita uma análise mais profunda sobre este trabalho.

Em setembro de 2004 foi lançado o projeto IPerG (Integrated Project on Pervasive Gaming) [IPe06]. Visa produzir experiências de jogo completamente novas e que sejam

intimamente associadas à vida cotidiana das pessoas. São realizadas pesquisas tanto sobre aspectos técnicos quanto de *gamedesign* a respeito de jogos pervasivos. Sendo assim, o projeto tem como objetivos facilitar a criação de jogos pervasivos, desenvolver projetos de jogos da área, compreender o público-alvo e compreender também o impacto social de tais jogos. O IPerG compreende vários subprojetos, dos quais dois estão mais relacionados a este presente trabalho: o MMRO e o *Crossmedia*.

O subprojeto *Massively Multiplayer Reaching Out* (MMRO) tem como objetivo o desenvolvimento de jogos inovadores que procurem integrar infra-estruturas técnicas de computação móvel, serviços baseados em localização e realidade virtual, com tradicionais MMOGs. Em sua descrição, o subprojeto MMRO propõe a idéia de dois ambientes ou "espaços" distintos, porém interconectados: um relativo ao mundo real e outro ao mundo virtual. Nesta abordagem, a estrutura de um jogo sugere que ações realizadas em diferentes subespaços (subconjunto do mundo virtual ou real) têm influência simultânea nos demais subespaços do jogo. Na proposta MMRO, esta característica cria um conceito chamado então de trans-realidade (*trans-reality*), em que o mundo virtual e o real são sintetizados em um único ambiente. Por exemplo, o deslocamento do usuário no mundo real representaria o deslocamento de seu avatar no mundo virtual. Além disso, são apresentadas idéias para o tratamento do eventual problema de sincronização entre o estado dos jogadores, principalmente localização, entre os dois mundos real e virtual.

O subprojeto *Crossmedia* visa desenvolver um jogo multiplataforma multiusuário, o *Epidemic Menace*. Nele, os jogadores representam profissionais da área médica que precisam salvar a humanidade de um vírus mutante criado por um cientista vilão. Eles se dividem em duas equipes com o objetivo de destruir o vírus enquanto apenas o campus Birlinghoven do instituto Fraunhofer FIT está contaminado, evitando que ele se espalhe e contamine toda a população mundial. Para tanto, os jogadores dispõem de diversas interfaces para jogar, incluindo aparelhos celular, sistemas móveis de Realidade Aumentada e contas de usuário para acessar o jogo pela Internet. Um primeiro protótipo de *Epidemic Menace* foi demonstrado e testado com oito jogadores em agosto de 2005 no próprio campus Birlinghoven, na Alemanha. Futuramente serão investigadas as possibilidades da utilização de TV interativa para a inclusão de mais jogadores e da integração de sistemas móveis de Realidade Aumentada de baixo custo.

Em outro trabalho do projeto, Koivisto [KW05] faz um levantamento sobre quais seriam as funcionalidades interessantes na integração da mobilidade em jogos massivos tradicionais. Por meio de entrevistas com um grupo de jogadores, foram levantadas seis funcionalidades:

- **Comunicação entre usuários fora do mundo virtual:** Canais de comunicação permitiriam que usuários pudessem trocar mensagens enquanto utilizassem dispositivos móveis;
- **Notificação de Eventos:** O uso de dispositivos móveis permitiria o envio de informações sobre o andamento do jogo, mesmo quando estes estivessem fora do mundo virtual;
- **Possibilidade de jogabilidade assíncrona:** Realização de tarefas que não demandassem interação em tempo-real, como desenvolvimento de habilidades do personagem ou compra e venda de objetos virtuais em um possível sistema de leilões do jogo;
- **Possibilidade de jogabilidade síncrona:** Permitir ao usuário orientar seu personagem a realizar certas ações no mundo virtual;
- **Participação Passiva:** Permitir que certas decisões de interesse global do mundo virtual possam ser tomadas através de um sistema de votação ou *ranking*, no qual também possam opinar os usuários com dispositivos móveis;
- **Realidade Paralela:** Fazer com certas ações do mundo real influenciem o mundo virtual.

3.3 PM³G

A partir dos trabalhos iniciais citados na subseção anterior, está sendo desenvolvida no Centro de Informática da UFPE uma tese de doutorado com o tema de Suporte a Pervasividade em Jogos Massivamente Multiusuário. Como explicado na seção 1, esta tese de doutorado motivou o presente trabalho de graduação. Desta forma, nesta seção serão explicados maiores detalhes de sua concepção, especificação e implementação.

A proposta do trabalho PM³G é criar um cenário onde jogos multiusuário possam ser acessados utilizando tanto computadores pessoais como dispositivos móveis. São ainda consideradas oportunidades específicas da computação móvel para acrescer características

pervasivas a tais jogos, de forma que o usuário possa estar sempre conectado ao mundo virtual do jogo.

Deste modo, é objetivo do trabalho a criação de uma infra-estrutura para amenizar o esforço exigido na implementação destes jogos, reduzindo custos e tempo de desenvolvimento. Tal infra-estrutura é constituída de um *framework* e um conjunto de serviços. O *framework* oferece uma API para a criação de jogos multiusuário, abstraindo aspectos da comunicação entre o servidor e os clientes e detalhes da simulação do jogo pelo servidor. Os serviços, por sua vez, formam uma camada de *middleware* para o suporte à pervasividade.

Para melhor compreensão da infra-estrutura e funcionalidade do *middleware* PM³G, são apresentados os cenários de utilização do *middleware*, que descrevem a forma como que os usuários interagem com o jogo PM³G. Em seguida, é descrita a visão de elementos que compõem o jogo, por meio de um modelo de aplicação. O modelo de programação é descrito na subseção seguinte e, por último, os serviços são listados e explicados.

3.3.1 Cenários de utilização

Os cenários de utilização propostos pelo trabalho exploram características pervasivas em jogos multiusuário e consideram diversos aspectos técnicos – como a atual infra-estrutura de telefonia celular e os dispositivos móveis disponíveis no mercado –, de forma a garantir sua viabilidade a curto prazo.

Em um cenário possível, o jogador acessa o jogo utilizando um computador pessoal com alto poder de processamento e conexão à Internet de alta largura de banda (cenário 1 da figura 3). Para tanto, ele pode estar em sua casa ou em um estabelecimento próprio para jogos de computadores em rede (*lan house*), por exemplo. Ele será capaz, portanto, de interagir em tempo real com o jogo e ter uma percepção detalhada dos eventos que ocorrem.

Entretanto, em determinadas situações torna-se inviável ao jogador ter acesso a computadores pessoais. Isto pode acontecer quando o jogador está, por exemplo, na fila do banco ou no ônibus. Nestas condições, é possível ao jogador acessar o jogo através de seu telefone celular utilizando a rede de telefonia móvel (cenário 2 da figura 3). Este cenário inviabiliza a interação com o mundo do jogo da forma que ocorre no cenário 1, mas permite

que o jogador possa realizar interações mais simples e obter informações resumidas a respeito do seu personagem ou do mundo do jogo.

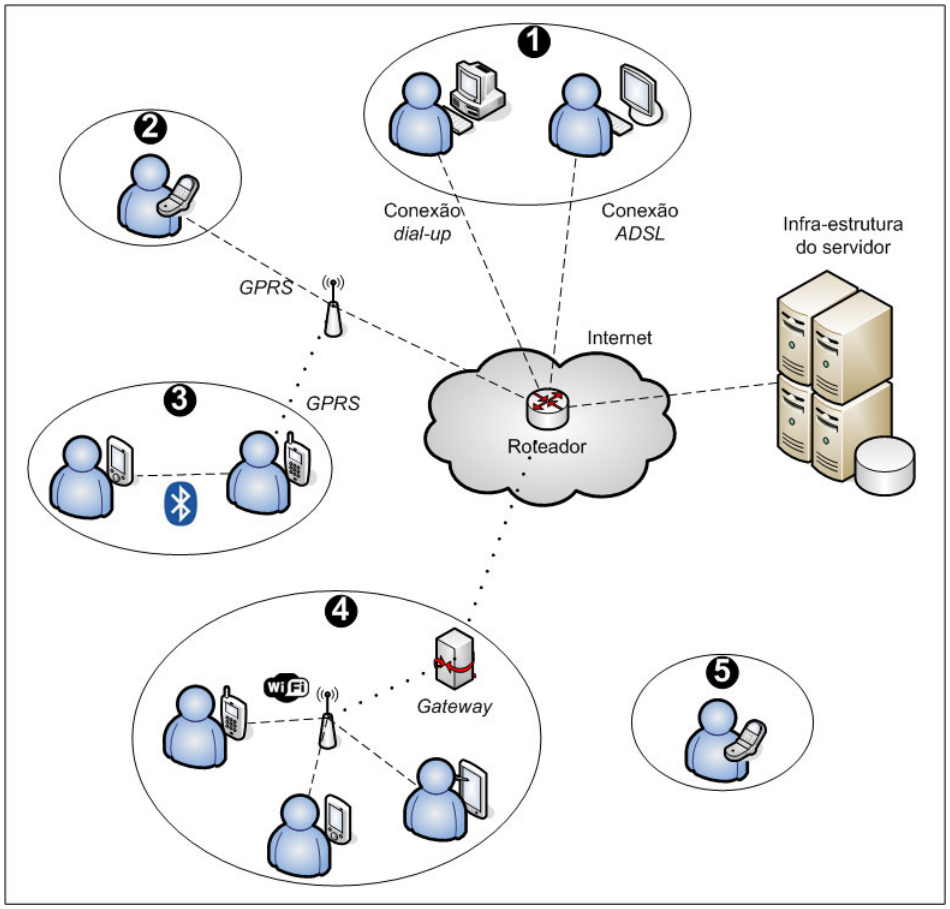

Figura 3 - Cenários de utilização de um jogo PM³G

Jogadores também poderiam utilizar seus dispositivos móveis para criar uma rede *ad-hoc* e disputar partidas desconectadas do servidor do jogo (cenário 3 da figura 3). Bastaria a eles estarem suficientemente próximos e possuir aparelhos que permitissem este tipo de comunicação, como telefones celulares com suporte a *Bluetooth*. Ao final de uma sessão de jogo entre os participantes, algum deles se conectaria ao servidor do jogo pela rede de telefonia e informaria ao servidor do jogo o resultado da sessão de jogo realizada.

Caso o usuário se encontre em um local que disponibilize acesso à internet através de um ponto de acesso *Wi-Fi* e possua um dispositivo habilitado com esta tecnologia, ele poderia acessar o jogo utilizando tal conexão (cenário 4 da figura 3). Esta forma de acesso permitiria uma maior interatividade com o jogo do que a conexão via rede da operadora do celular. Além disso, poderia haver no local um servidor em que parte do mundo virtual do jogo esteja simulado, possibilitando a todos os demais jogadores que estivessem utilizando o mesmo

ponto de acesso participar de uma sessão de jogo local e temporária, cujos resultados obtidos teriam efeito sobre o mundo virtual persistente do jogo.

Finalmente, considera-se ainda a possibilidade de interação com o jogo mesmo que o usuário não disponha de nenhuma forma de acesso *online* ao jogo – o que poderia ocorrer, por exemplo, caso a operadora de telefonia móvel estivesse temporariamente indisponível para o seu dispositivo celular (cenário 5 da figura 3). Neste caso, seria possível jogar algum módulo *offline* do jogo, realizando determinadas atividades cujos resultados seriam posteriormente sincronizados com o servidor do jogo.

As participações dos jogadores ocorrem através de sessões de interação com o mundo virtual. Uma sessão caracteriza-se por um intervalo de tempo em que um jogador interage com o jogo a partir de um dispositivo específico, quer seja um computador pessoal ou um dispositivo móvel. Toda sessão é iniciada através de um processo de autenticação do jogador com os dados de identificação do usuário e sua senha de acesso. Quando o jogador migra para outro dispositivo, a sessão é finalizada e uma nova sessão é iniciada, permitindo a continuidade das ações do avatar.

3.3.2 Modelo de aplicação

O modelo de aplicação PM³G leva em conta que as características de um jogo multiplataforma multiusuário são mais bem exploradas quando o mundo virtual é persistente. Isto permite que os jogadores possam se desconectar do jogo e voltar a jogar na mesma partida, mantendo as informações de seus personagens. Jogos com mundo virtual persistente são comuns entre os 3MOGs, como já explicado na seção 2.2.5.

Desta forma, o modelo PM³G baseia-se nos padrões da maioria dos jogos massivos existentes. Nestes jogos, o mundo virtual é representado como uma grande região habitada por centenas ou milhares de jogadores, cada um sendo representado por um avatar. A dinâmica do jogo tipicamente envolve a realização de missões pelos jogadores – seja individualmente ou em grupos –, demandando deslocamentos por diferentes partes do mundo virtual, interações e possíveis disputas com outros jogadores, além de busca por itens virtuais e acúmulo de experiência (geralmente na forma abstrata de habilidades adquiridas).

Baseado nesta visão e em modelos arquiteturais para MMOGs ([Cec05] e [KLXH04]), são extraídos os seguintes conceitos-chaves para este tipo de aplicação: terreno, objeto, mundo, jogador, personagens automatizados e áreas de interesse. O modelo de aplicação

PM³G estende este modelo criando quatro novos conceitos: contexto de execução, percepção, interação e minimundo. A seguir, é dada uma breve descrição dos conceitos que são relevantes a este presente trabalho de graduação:

- **Terreno:** representa o conjunto de informações imutáveis que compõem a paisagem do mundo virtual, influenciando o deslocamento dos jogadores.
- **Objeto:** é o elemento básico que compõe o conjunto de informações dinâmicas do mundo virtual. Em outras palavras, um objeto é qualquer informação que pode ser adicionada, copiada, alterada ou removida do mundo virtual. Exemplos típicos de objetos são comidas, armas ou ferramentas.
- **Mundo virtual (ou simplesmente "mundo"):** é o sistema de regras ou leis que implementam o espaço virtual onde estão localizados os objetos. Objetos localizados no mesmo mundo podem interagir entre si e normalmente as interações entre objetos ocorrem com maior freqüência quando estes objetos estão próximos no mundo virtual.
- **Jogador:** representa o usuário do jogo, que interage com o mundo ao exercer controle sobre objetos. Cada jogador tem um estado associado, composto pelas informações de seu avatar, como habilidades, energia vital e posses.
- **Personagens automatizados:** também conhecidos por NPCs[5], são personagens cujas ações não são controladas por um jogador. Podem representar inimigos, aliados ou simplesmente espectadores (*bypassers*). Assim como os jogadores, estes elementos também possuem um estado e um avatar associado.
- **Área de interesse:** é um subconjunto do mundo virtual com o qual o jogador pode interagir em um dado instante (ver seção 2.3.2).
- **Contexto de execução:** relaciona-se com a infra-estrutura de acesso de um jogador ao mundo virtual, incluindo o tipo de rede e o dispositivo utilizado para acessar o jogo.
- **Percepção:** é a forma com que o jogo é apresentado ao jogador. É influenciada diretamente pelo contexto de execução do jogador naquele instante.
- **Interação:** é a maneira como o jogador realiza suas ações no mundo virtual. Depende do contexto de execução e da percepção do jogador em um dado momento.

[5] Sigla de *Non-player Character*

3.3.3 Modelo de programação

O *framework* em desenvolvimento é composto por uma API para a criação de um jogo multiplataforma multiusuário. Simplificadamente, o jogo consiste tanto no servidor do jogo como na aplicação cliente. É oferecido um suporte à comunicação entre o servidor e as aplicações clientes, de forma que o desenvolvedor do jogo possa abstrair tais detalhes. A infra-estrutura de comunicação foi baseada na arquitetura do servidor de jogo em [BBV03]. Para o servidor, há também o suporte ao tratamento de eventos do jogo e à simulação do jogo, embora não haja suporte à persistência do estado do jogo. Para a aplicação cliente, é oferecida uma API de programação para aplicações em *desktops* (com Java SE) e dispositivos móveis (Java ME).

Para a especificação da API, foram utilizados os conceitos-chave explicados na seção anterior. A partir destes conceitos, e baseando-se na arquitetura proposta em [Ale02], foram definidas algumas entidades genéricas, de forma que o desenvolvimento dos jogos possa ser estruturado e padronizado, mas sem restringir as possibilidades de criação para diferentes tipos de jogos. As figuras 4 e 5 apresentam diagramas de classes em UML simplificado para algumas destas entidades. Visto que o a lógica dos serviços se baseia nestas entidades, serão descritas resumidamente algumas delas a seguir. Maiores detalhes da implementação do *middleware* podem ser obtidos em [Tri06b] [Tri06c].

- **`GameObject:`** representa o conceito de Objeto do modelo de aplicação. Possui uma posição, que indica a sua localização no mundo. Possui também uma dimensão que diz respeito ao tamanho do objeto no mundo virtual do jogo, e independe da forma com que o objeto é apresentado ao jogador.
- **`Performer:`** é o objeto que realiza uma ação. Pode ser um NPC ou um personagem do jogador. Possui uma máquina de estados finita que controla a ação realizada por este objeto em um dado momento da simulação.
- **`Actor:`** é uma especialização de `Performer` que representa o personagem controlado por um jogador. Possui uma fila de eventos a serem realizados, criados por comandos do jogador e executados a cada passo da simulação do jogo.
- **`FiniteStateMachine:`** trata-se da máquina de estados de um `Performer`. Armazena seu estado ativo e os demais estados que o objeto pode assumir, e também controla suas mudanças de estado.

- **`State:`** representa cada estado que um `Performer` pode assumir, definindo o comportamento do mesmo.

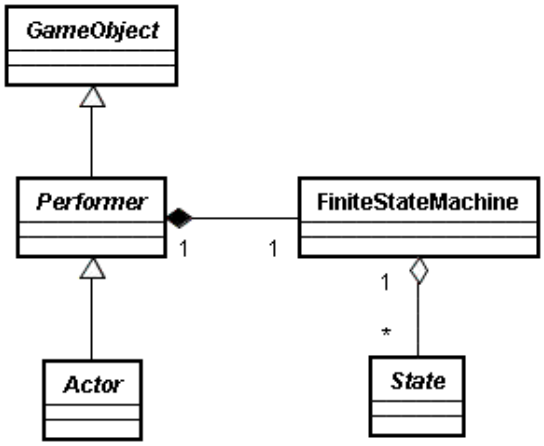

Figura 4 - Diagrama de classes de GameObject, Performer, Actor, FiniteStateMachine e State.

- **`GameEvent:`** trata-se de um evento de jogo trocado entre uma aplicação cliente e o servidor. Caso seja um evento enviado pelo cliente, terá um jogador associado. Por outro lado, terá um ou mais destinatários se for um evento enviado pelo servidor. Possui um código informando o tipo de evento, e também um conjunto genérico de informações que são tratadas dependendo deste tipo.
- **`AreaOfInterest:`** representa a área de interesse do jogador, e possui três atributos indicando seu conteúdo: `actors`, `items` e `performers`. O primeiro corresponde aos avatares dos jogadores; `items` são os objetos com os quais os atores podem interagir; e `performers` tratam-se dos NPCs.
- **`Player:`** representa um jogador, com suas informações pessoais – *login*, senha e endereço de e-mail, por exemplo –, informações do seu contexto e da área de interesse do seu avatar.

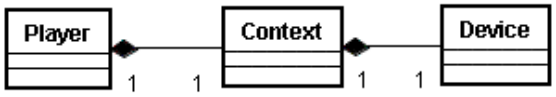

Figura 5 - Diagrama de classes de Player, Context e Device.

- **`Context:`** representa o contexto de um jogador, e possui o atributo `Device`, correspondente ao dispositivo que o jogador utiliza no momento.

- **`Device:`** possui um código correspondente ao modelo do dispositivo e um conjunto genérico de propriedades, que pode dizer respeito, por exemplo, à resolução da tela do dispositivo, ao idioma escolhido pelo jogador etc.

3.3.4 Serviços

Como visto na seção 2.4, *middlewares* de suporte a jogos multiusuário atacam questões comuns ao desenvolvimento destes jogos, relacionadas tanto a requisitos funcionais – como gerência de contas, autenticação e tarifação de jogadores – como requisitos não-funcionais, como escalabilidade e tolerância a falhas. Tais serviços são tipicamente construídos sobre uma base de serviços de suporte, como os de comunicação e persistência.

A proposta de suporte a um jogo PM³G amplia os serviços de um jogo massivo tradicional, agregando serviços de suporte relativos aos conceitos específicos do modelo de aplicação descrito na seção 3.3.2. São criados seis novos serviços, descritos a seguir:

- **Serviço de Gerenciamento de Contexto:** realiza o monitoramento do contexto de execução de um jogador, para que adaptações necessárias para a participação do jogador no jogo possam ser realizadas pelos demais serviços. O fornecimento de informações sobre a mudança de contexto do jogador é realizado pelas aplicações que ele utilizar, ao ser realizada a migração de sessões entre dispositivos, mudança de localização física ou alteração de preferências pessoais do jogador.
- **Serviço de Adaptação de Conteúdo:** tem como função transformar informações relevantes da área de interesse de um jogador para um formato adequado ao seu contexto de execução, seguindo o conceito de percepção do modelo de aplicação PM³G.
- **Serviço de Adaptação de Interação:** transforma as diferentes formas de entradas dos jogadores (possivelmente em contextos diversos) em ações padronizadas dentro da simulação do jogo.
- **Serviço de Notificação de Eventos:** informa aos jogadores sobre acontecimentos de seu interesse, principalmente quando estes jogadores estiverem em um contexto móvel. Permite que o jogador esteja seja sempre presente, sendo notificado através de diferentes meios de comunicação, como mensagens SMS ou *e-mail*.

- **Serviço de Localização Física:** tem por função rastrear a posição do jogador para que possa ser utilizada no suporte à inclusão de objetos virtuais no mundo real ou no auxílio à formação dos mini-mundos.
- **Serviço de Integração de Mini-Mundos:** responsável pela integração do jogo massivo com cenários formados pelo conceito de mini-mundo do modelo de aplicação PM³G. Mini-mundos são cenários paralelos com interação direta entre jogadores móveis, através de redes espontâneas ou WLANs. Estes cenários podem ser compreendidos como partidas isoladas entre jogadores móveis, porém com conseqüências no jogo massivo. Assim, por meio deste serviço cada partida paralela iniciada repassa ao jogo informações de seus jogadores e os resultados finais da partida.

CAPÍTULO 4

Serviços especificados e implementados

Especificar, implementar e testar dois serviços do *middleware* PM³G são os principais objetivos deste trabalho de graduação. A partir dos cenários propostos pela tese de doutorado de Trinta [Tri06a], de algumas entidades do *framework* especificadas e do Serviço de Gerenciamento de Contexto criado, foram definidos os seguintes requisitos para o projeto do Serviço de Adaptação de Conteúdo e do Serviço de Adaptação de Interação:

- **Facilidade de uso:** ambos os serviços devem ser fáceis de utilizar e de configurar pelo desenvolvedor do jogo;
- **Flexibilidade:** os serviços devem ser o mais genéricos possíveis, não se restringindo a uma categoria específica de jogos multiusuário;
- **Extensibilidade:** os serviços devem ser extensíveis, permitindo a qualquer desenvolvedor que utilize o *middleware* aperfeiçoar ou especializar suas funcionalidades se assim desejar;

Esta seção descreve a principal contribuição deste trabalho, explicando para cada serviço as entidades especificadas, a arquitetura proposta e as decisões de implementação adotadas.

4.1 Serviço de Adaptação de Conteúdo

A utilização de diferentes plataformas traz a reboque o problema da heterogeneidade das características de cada uma delas. Diferentes tamanhos de tela, capacidade de memória e de processamento, largura de banda e latência da conexão com a rede são alguns exemplos. Este problema demanda que as informações enviadas ao jogador sejam adaptadas de acordo

com o seu dispositivo utilizado, sendo apresentadas da forma mais adequada. Desta forma, o Serviço de Adaptação de Conteúdo é o responsável por esta adaptação.

O serviço foi concebido para ser utilizado da seguinte forma: conhecendo o conteúdo da área de interesse do jogador e sabendo qual o seu contexto de execução no momento, o serviço é requisitado pelo servidor do jogo para que as informações a serem enviadas ao jogador sejam adaptadas para o seu dispositivo. Tais informações são então retornadas ao jogo e este as repassa ao jogador.

4.1.1 Modelagem

Para definir como cada dispositivo determinaria a adaptação de conteúdo a ser realizada, foi criado o conceito de *estratégia de adaptação*. Uma estratégia de adaptação é associada a cada dispositivo, e determina quais transformações são realizadas nas informações a serem enviadas ao jogador.

Para modelar este serviço, foi observado que o conjunto de dispositivos abordados pode ser ordenado no que diz respeito à quantidade de informações recebidas pelo jogador, característica que se relaciona com a sensação de imersão provocada no mesmo. Com isto em mente, foi criado um atributo na classe `Device` (figura 5) que diz respeito ao nível de abstração do dispositivo. Quanto maior for este valor, menos informações o jogador que o utiliza irá receber. Um PC, por exemplo, pode ser considerado o dispositivo com o menor nível de abstração em um determinado jogo. Visto que o jogador que utiliza um computador pessoal deseja uma maior sensação de imersão no jogo, as informações precisam ser apresentadas ao jogador de maneira detalhada, o que exige muita troca de dados entre a aplicação cliente e o servidor. Já um aparelho celular de limitados recursos computacionais oferece uma mínima sensação de imersão ao jogador; sendo assim, são poucas as informações recebidas pela aplicação cliente.

Cada elemento da área de interesse também deve conter uma informação a respeito da sua importância, para que o serviço saiba quais objetos podem ser descartados e quais devem ser enviados ao jogador. Para isto, foi incluído na classe `GameObject` (figura 4) o atributo `priority`. Esta informação permite ao desenvolvedor do jogo criar categorias como, por exemplo, essencial, importante ou opcional, para cada ator, item ou NPC contido em uma área de interesse.

Diversos tipos de processamento podem ser realizados sobre as informações de conteúdo enviadas ao jogador. Neste trabalho, estes processamentos são divididos em duas categorias: *processamentos lógicos* e *processamentos de apresentação*. A primeira categoria engloba procedimentos que modificam o conteúdo da área de interesse, como, por exemplo, a quantidade de objetos ou a posição deles no mundo virtual. Por sua vez, processamentos de apresentação preparam os dados para serem enviados à aplicação cliente. Dependendo da forma com que estas informações são enviadas e também do dispositivo utilizado, pode ser necessário que estas informações recebidas pela aplicação cliente sejam apresentadas ao jogador com a menor quantidade de processamento local possível.

Cada forma de processamento é representada por um *módulo de adaptação de conteúdo*. Desta maneira, os módulos são classificados em *módulo lógico* ou *módulo de apresentação*, que implementam, respectivamente, a interface `LogicModule` ou `PresentationModule` (figura 8). A princípio, foram definidos os seguintes módulos lógicos:

- **Módulo de filtragem de informações (`FilteringModule`):** baseando-se no nível de abstração do dispositivo e na prioridade de cada elemento da área de interesse, este módulo é responsável por descartar os elementos que não precisem ser enviados ao jogador. Isto traz duas vantagens: a primeira é a redução do tráfego na rede, o que pode ser importante para jogadores utilizando algum tipo de conexão cuja cobrança é dada pela quantidade de bytes transmitidos. Outra vantagem é a menor carga de processamento e memória exigidos do dispositivo, já que são menos objetos a serem processados pela aplicação cliente.
- **Módulo de escala bidimensional (`ScalingModule`):** ajusta as posições dos objetos de uma área de interesse (bidimensional) baseando-se na resolução do dispositivo utilizado. Apesar de ser uma operação simples, o fato de ser realizada pelo serviço e não no dispositivo cliente diminui a carga de processamento neste último.
- **Módulo de *tiling* (`TilingModule`):** adapta as posições dos objetos de uma área de interesse (bidimensional) para que possam ser apresentados como em jogos baseados em *tiles* (*tile-based games*). Nestes jogos, um mapa é formado por blocos bidimensionais de igual tamanho, que o preenchem completamente e sem sobreposição. Logo, a posição de qualquer objeto apresentado neste mapa deve assumir uma quantidade limitada de valores, que podem ser mapeados por uma matriz com N linhas e M colunas.

- **Módulo de projeção tridimensional (`ProjectingModule`):** realiza a projeção tridimensional das posições dos objetos para valores bidimensionais. Isto é muito útil em jogos 3D que precisem ser apresentados em dispositivos cujas características inviabilizem a utilização de gráficos tridimensionais (como, por exemplo, o tamanho da tela ou a capacidade de processamento). A posição dos objetos da área de interesse enviada ao jogador será definida, portanto, de maneira bidimensional.

No serviço, para cada estratégia de adaptação, podem ser utilizados quantos módulos lógicos for necessário. Apesar de somente quatro terem sido concebidos, o serviço é extensível para que outros possam ser criados e acoplados. Desta forma, o resultado da transformação realizada por um módulo lógico serve de entrada para o módulo seguinte.

Os módulos lógicos acima descritos possuem a propriedade de que, não importa a ordem em que eles processam um mesmo conjunto de dados de uma área de interesse, o resultado será sempre o mesmo. Contudo, o custo de processamento pode variar dependendo desta ordem. Por exemplo, caso o módulo de escala bidimensional trate o conteúdo de uma área de interesse antes do módulo de filtragem, será ajustada a posição de elementos que possivelmente serão descartados no módulo seguinte. Desta maneira, percebe-se que é mais vantajoso ordenar os dois módulos para que o de filtragem seja executado antes do de escala.

Duas recomendações, portanto, são feitas a partir destas observações. A primeira é que novos módulos lógicos criados e adicionados ao serviço mantenham a propriedade de garantia do mesmo resultado final independentemente da ordem em que os módulos realizam o processamento. A segunda recomendação é que o funcionamento dos módulos seja compreendido pelo desenvolvedor do jogo para ordená-los da melhor maneira, se quiser utilizá-los de maneira otimizada.

Por sua vez, em relação aos processamentos de apresentação, foram definidos os seguintes módulos:

- **Módulo de apresentação textual:** baseado no idioma escolhido pelo jogador, este módulo cria um texto informando resumidamente o conteúdo da área de interesse. O idioma é armazenado como uma propriedade do dispositivo, obtido a partir do seu contexto. Desta forma, são informados quantos atores, itens e NPCs estão presentes, sendo também incluída a representação textual de cada um deles. É permitido ao desenvolvedor do jogo escolher como é criada a representação textual de cada elemento na criação das entidades que herdam de `GameObject`.

- **Módulo de apresentação padrão:** neste módulo todas as informações resultantes da área de interesse são serializadas de uma maneira padrão, de acordo com um protocolo que deve ser de conhecimento do desenvolvedor do jogo.

Cada módulo de apresentação define inteiramente a forma com que o conteúdo da área de interesse é enviado ao jogador. Assim, faz sentido que somente um módulo de apresentação seja utilizado em cada estratégia de adaptação. Vale ressaltar que, se o desenvolvedor preferir, pode ser adicionado ao serviço outro módulo de apresentação para que os dados sejam serializados da forma desejada.

É importante ainda observar que o caráter genérico do serviço exige que as estratégias de adaptação a serem utilizadas pelo jogo devem ser previamente definidas pela simulação do jogo. Deste modo, na inicialização do servidor do jogo, as estratégias são criadas e repassadas ao serviço.

Finalmente, para evitar uma degradação do desempenho do sistema na adaptação de conteúdo para jogadores com atualização freqüente do estado do jogo, foi utilizada a idéia de uma *cache* para armazenar localmente a estratégia de adaptação do jogador estacionário. Desta forma, o serviço oferece também uma forma de obter a estratégia de adaptação para um determinado dispositivo, de forma que a simulação de jogo possa obter a adaptação de conteúdo de um jogador sem a necessidade de chamar remotamente o serviço a cada ciclo de jogo.

4.1.2 Diagrama de seqüência

O fluxo de mensagens da operação `adaptContent` ocorre de acordo com a figura 6. Ela possui dois parâmetros: a área de interesse do jogador e o seu contexto. Logo que a operação é chamada a partir da simulação do jogo, o serviço delega a lógica da operação para o `AdaptationStrategyManager`, objeto que armazena todas as estratégias de adaptação previamente definidas. Este verifica qual a estratégia de adaptação que será utilizada, baseando-se no contexto do jogador, especificamente sobre o seu dispositivo. Para cada módulo lógico da estratégia de adaptação (se houver algum), a área de interesse sofre um processamento até ser serializada pelo módulo de apresentação da estratégia. O resultado é, então, retornado à simulação do jogo.

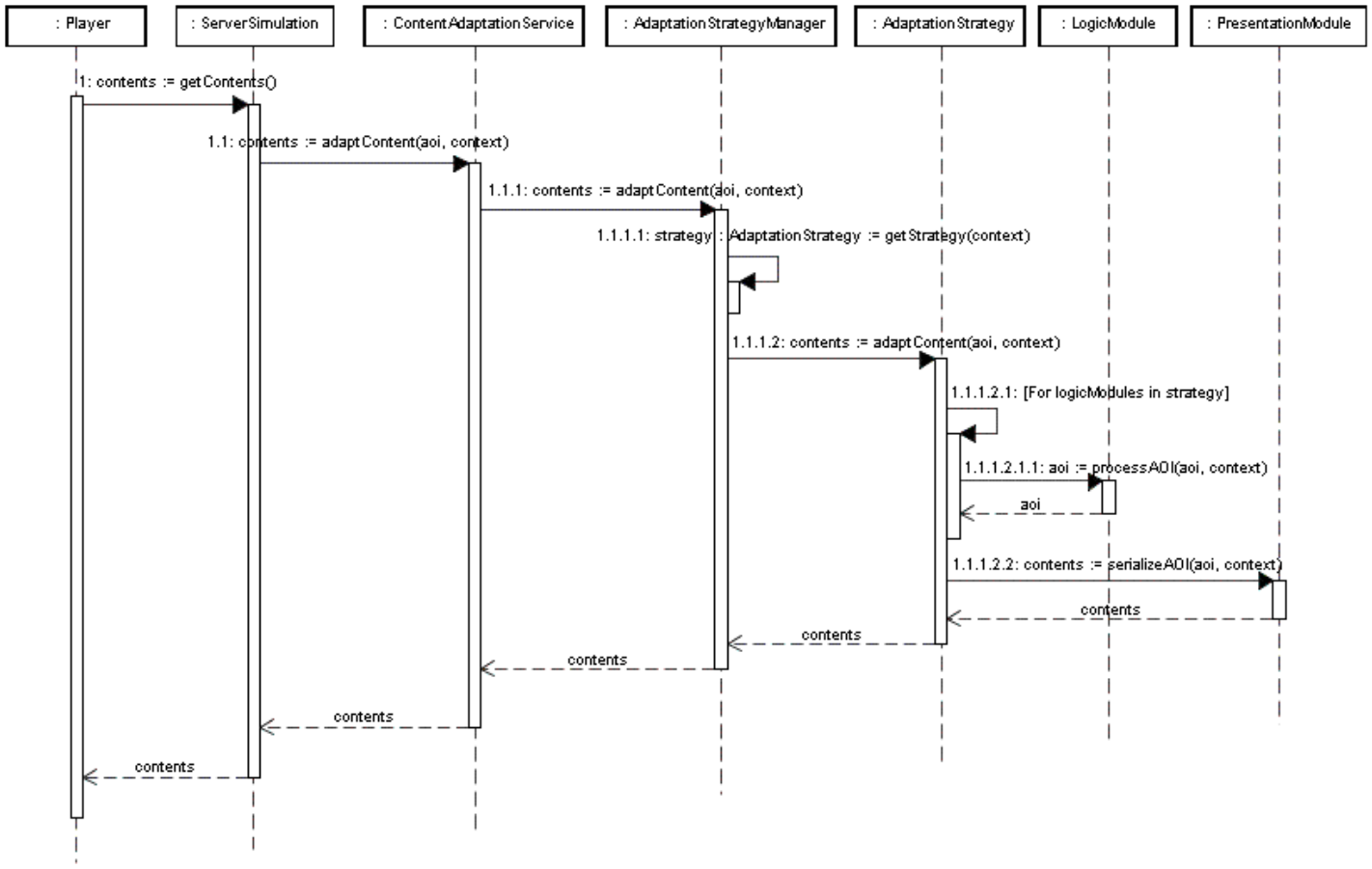

Figura 6 - Diagrama de seqüência da operação adaptContent do Serviço de Adaptação de Conteúdo.

A operação `getStrategy` pode ser utilizada pela simulação de jogo para realizar a adaptação de conteúdo de jogadores estacionários, cuja taxa de atualização do estado do jogo é muito freqüente. Desta forma, a simulação obtém a estratégia de adaptação do jogador e a armazena em uma *cache*, fazendo com que ela realize a adaptação localmente da mesma maneira que é feita no serviço. O fluxo de mensagens é mostrado na figura 7.

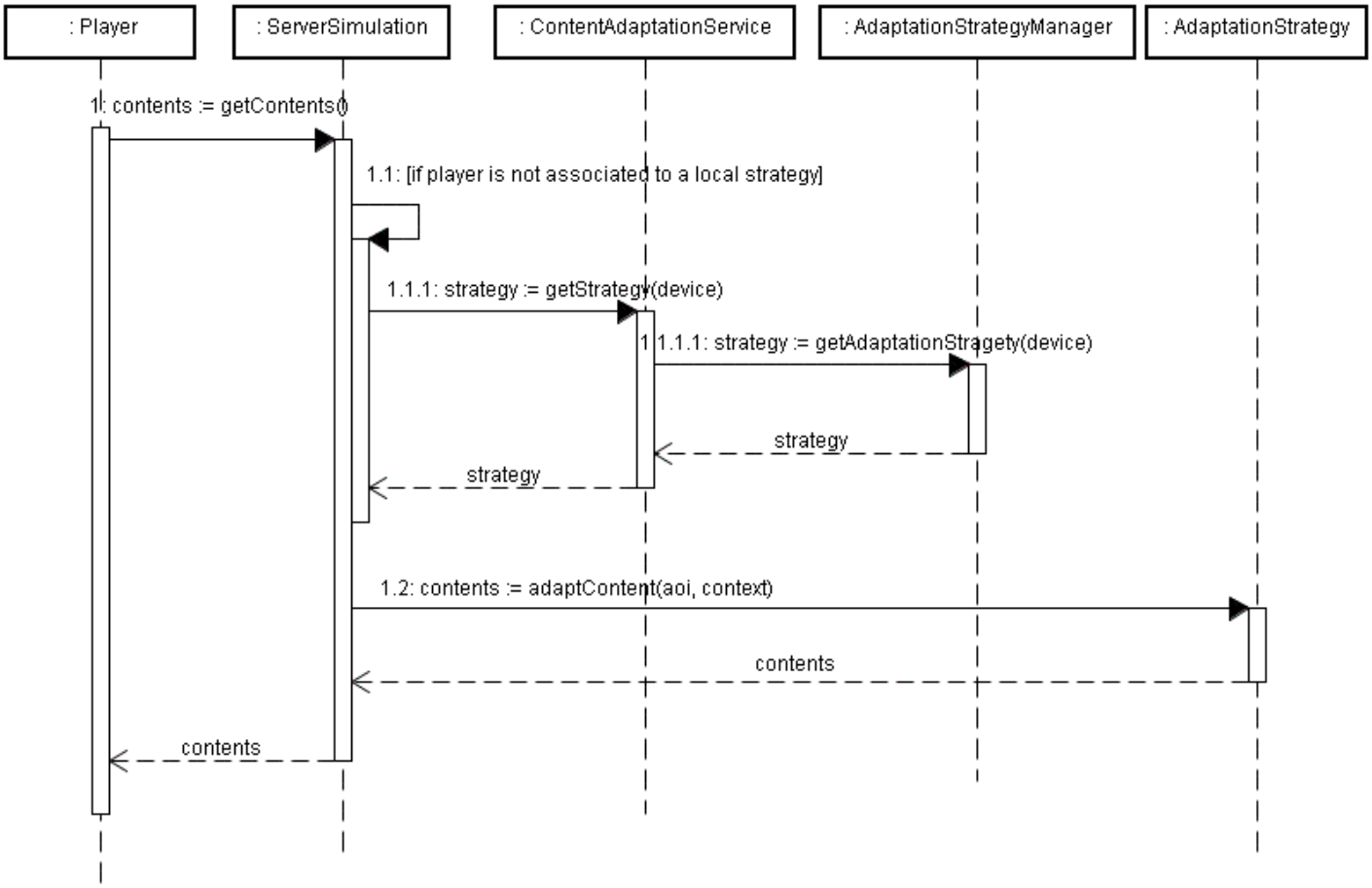

Figura 7 - Diagrama de seqüência da operação getStrategy do Serviço de Adaptação de Conteúdo.

4.1.3 Arquitetura proposta

A figura 8 apresenta a arquitetura proposta para o Serviço de Adaptação de Conteúdo como um diagrama de classes em UML [OMG03].

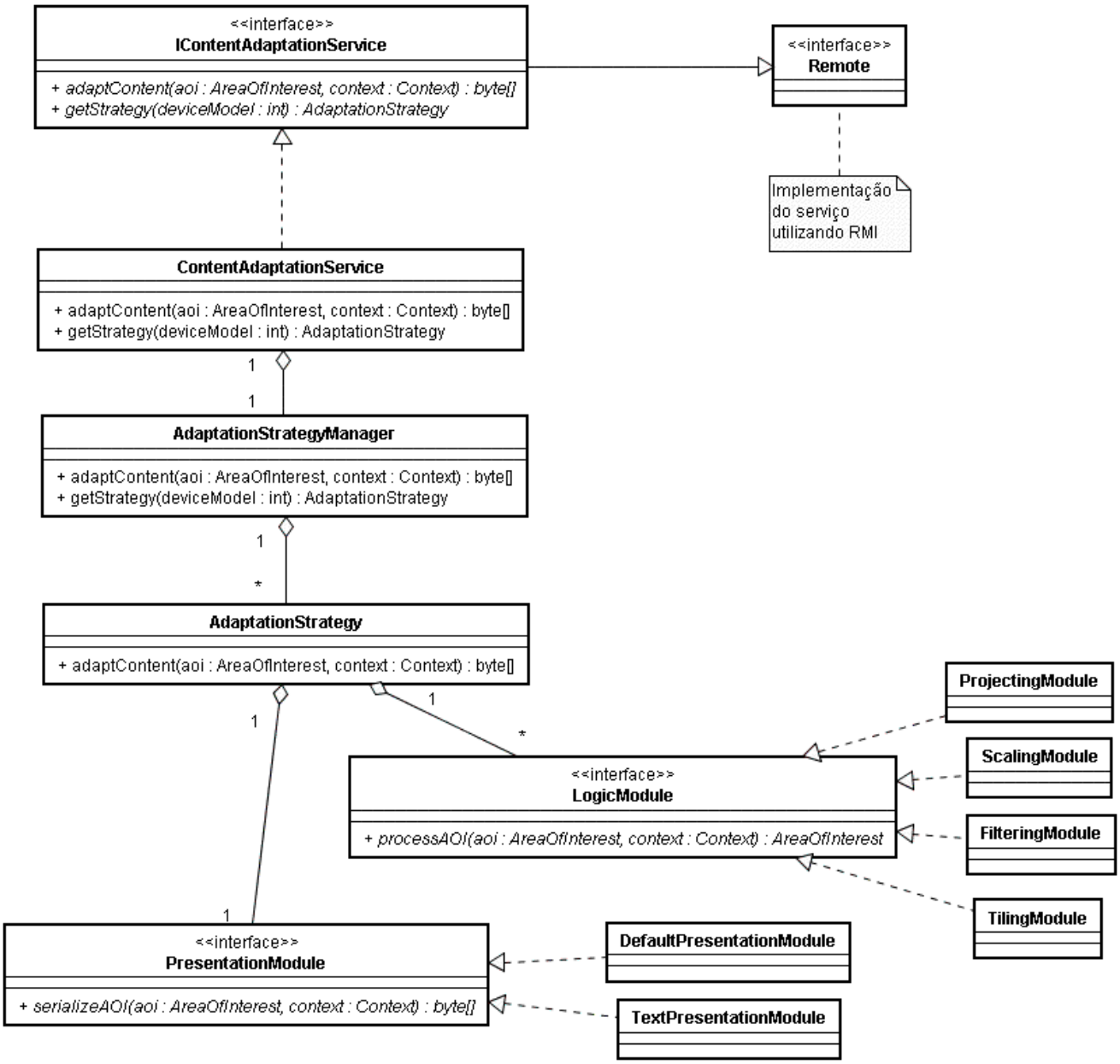

Figura 8 - Diagrama de classes do Serviço de Adaptação de Conteúdo.

4.2 Serviço de Adaptação de Iteração

O uso de diferentes dispositivos permite que as entradas das ações dos jogadores móveis e estacionários sejam realizadas de formas diferentes. Porém, independentemente do contexto de execução do jogador, estas diferentes ações devem ser mapeadas para ações comuns dentro do mundo virtual do jogo. O jogo multiplataforma deve permitir que jogadores estacionários e móveis interajam diretamente no mundo virtual realizando ações equivalentes.

O Serviço de Adaptação de Interação propõe-se a transformar as diferentes formas de entradas dos jogadores em ações padronizadas dentro de cada área de interesse do jogo. Em um contexto estacionário a atuação do serviço pode ser dispensável, uma vez que o jogador estará efetivamente controlando as ações do seu personagem. No contexto móvel, por sua vez, a adaptação de interação é imprescindível, visto que a limitação da interface homem-máquina dos dispositivos móveis faz com que as típicas ações baseadas em movimentações ágeis do avatar sejam inadequadas neste contexto. Sendo assim, as ações de um jogador móvel devem ser realizadas sem a necessidade de uma conexão constante com o jogo ou de respostas imediatas aos seus comandos, podendo o jogador disparar uma ação a ser executada no mundo virtual enquanto ele realiza outras atividades fora do jogo, como deslocamento físico ou atendimento de chamadas no aparelho celular.

No Serviço de Adaptação de Interação, estes requisitos são acatados através da idéia proposta por [Fox03]. As ações de jogadores móveis são realizadas no mundo virtual através de comandos de alto nível, que são interpretadas pelo jogo em um conjunto de ações concretas. Na realidade, esta idéia segue exemplos já existentes em jogos comerciais. Por exemplo, o jogo *Starcraft* [BE06b], permite que o jogador coloque tropas em estado de patrulha, onde em caso da presença de um adversário, as tropas atacam automaticamente. Este jogo permite também que tropas desloquem-se no mapa, através da indicação apenas de seu destino, sem que seja necessário um efetivo controle de seus avatares. O comportamento do personagem após o comando executado seria semelhante a um NPC, controlado por uma inteligência artificial.

4.2.1 Modelagem

Para modelar este serviço, foi necessário definir como seria a interação do usuário com o jogo em um dispositivo móvel de interface homem-máquina limitada. Uma solução

encontrada seria fazer com que o jogo enviasse ao jogador o conjunto de ações de alto nível possíveis do seu personagem. O jogador, então, informaria ao servidor do jogo a ação escolhida e este se responsabilizaria – utilizando o Serviço de Adaptação de Interação – a mapear o comando em ações padrões do jogo.

- **Ações do jogador móvel**

Para realizar a obtenção das ações possíveis de um avatar em um dado instante, convém definir quais aspectos do jogo influenciam este procedimento. Em primeiro lugar, foi definido que o estado do avatar é um ponto de partida para descobrir quais ações ele pode realizar. Afinal, cada ação pode ser enxergada como uma espécie de transição de estados; ela indica qual o comportamento que o personagem deverá assumir a partir daquele momento.

A figura 9 exemplifica esta idéia: nela, estão associados duas ações ao Estado 1 (Ação 1 e Ação 2), enquanto que a partir dos outros estados não é possível realizar nenhuma ação. As transições que saem dos estados 2 e 3 não são, portanto, definidas por comandos do jogador.

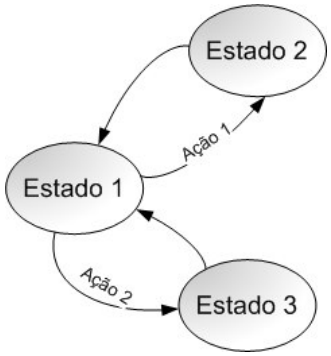

Figura 9 - Exemplo da relação entre estados e ações de um avatar.

Entretanto, é importante notar que não apenas o estado do jogador determina as suas ações. Dependendo do conteúdo da área de interesse, determinadas ações podem ser realizadas ou não. Por exemplo, um jogador não poderá realizar nenhuma ação de interação com itens virtuais caso não haja nenhum item virtual visível na sua área de interesse. Do mesmo modo, ele também não poderá interagir com outro personagem caso ele seja o único personagem em sua área de interesse naquele momento. Assim, foi definido que o conteúdo da área de interesse também influencia na obtenção das ações do jogador.

Com isto em mente, foi criada a entidade `Action`, que representa as ações de alto nível dos jogadores (figura 10). Esta entidade possui uma descrição textual, usada na exibição das ações para os jogadores, além do atributo `isRelatedToAOI`, que informa se a ação está

relacionada ao conteúdo da área de interesse do jogador. Esta informação é importante para permitir que algumas ações possam ocorrer sempre, independentemente do conteúdo da área de interesse do jogador. Um exemplo é uma ação para se desconectar do jogo, que a princípio pode ser realizada a qualquer momento.

Um objeto `Action` está sempre relacionado a um evento de jogo (`GameEvent`). Esta relação determina de maneira simples como o jogo irá interpretar a ação, pois um `GameEvent` é a forma padrão com que a simulação de jogo trata os eventos recebidos dos jogadores.

Uma ação, como já foi dito, pode estar intimamente relacionada a um elemento específico da área de interesse do jogador. Pode ser, por exemplo, um outro personagem com o qual deseja-se interagir, ou um item qualquer. Deste modo, deve ser possível ao jogador, no momento em que seleciona uma ação deste tipo, especificar o objeto do jogo ao qual a ação se refere. Isto é possível com o uso de uma lógica do serviço que, ao oferecer uma ação a um jogador, liste opções relativas àquela ação. Por causa disto, ao objeto `Action` é atribuído um conjunto de objetos `GameObject`, cada um representando uma opção daquela ação que pode ser escolhida pelo jogador.

Para saber se um objeto da área de interesse será considerado uma opção de uma determinada ação, foi criado o método abstrato `isRelatedToAction`, que deve determinar para cada ação do jogo se o objeto se relaciona ou não com ela.

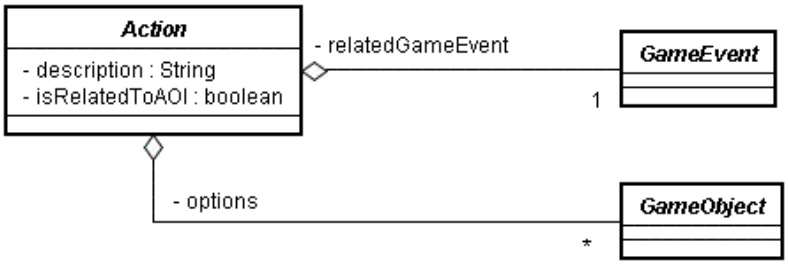

Figura 10 - Diagrama de classes de Action.

Vale ressaltar que o serviço, por ser genérico, não tem como identificar que ações podem ser realizadas para cada estado de um avatar. Na realidade, o jogo é que define quais são os seus estados e ações. Isto ocorre na inicialização do servidor do jogo, quando este mapeamento é definido e informado ao serviço.

- **Serviço de Adaptação de Interação e contexto de execução**

Foi visto na seção 4.1.1 que o Serviço de Adaptação de Conteúdo toma ciência do contexto de execução do jogador sempre que é chamado pela simulação do jogo. O Serviço de

Adaptação de Interação, por sua vez, foi modelado para que não precise desta informação a todo momento.

Cada avatar possui um conjunto de estados que pode assumir, como já foi explicado. O que foi definido durante a modelagem deste serviço é que cada estado está diretamente relacionado a um nível de abstração do contexto de execução. O atributo `abstractionLevel` de `Device`, além de ser utilizado pelo Serviço da Adaptação de Conteúdo para determinar a quantidade de informações enviadas ao jogador, passa a indicar o nível de abstração das suas ações.

Existem, na realidade, vários conjuntos de estados, para cada contexto que o jogador pode assumir no jogo. É desta forma que este serviço tem condições de saber quais ações o jogador poderá realizar para um dado contexto, baseando-se apenas no estado atual do seu avatar. Simplificadamente, caso o jogador esteja em um contexto móvel, seu avatar deverá assumir um entre os possíveis estados relacionados a este contexto. Se ele passar a jogar em um computador pessoal, o estado do seu avatar deverá refletir a mudança de contexto. Este processo também é realizado pelo Serviço de Adaptação de Interação. Ele deve saber, a partir do estado de um avatar e do novo contexto do jogador, qual o novo estado que será assumido.

A figura 11 exemplifica a utilização de diferentes conjuntos de estados para cada contexto de execução, bem como associações entre estados de diferentes contextos, necessárias para atualizar o estado do avatar na mudança de contexto do jogador.

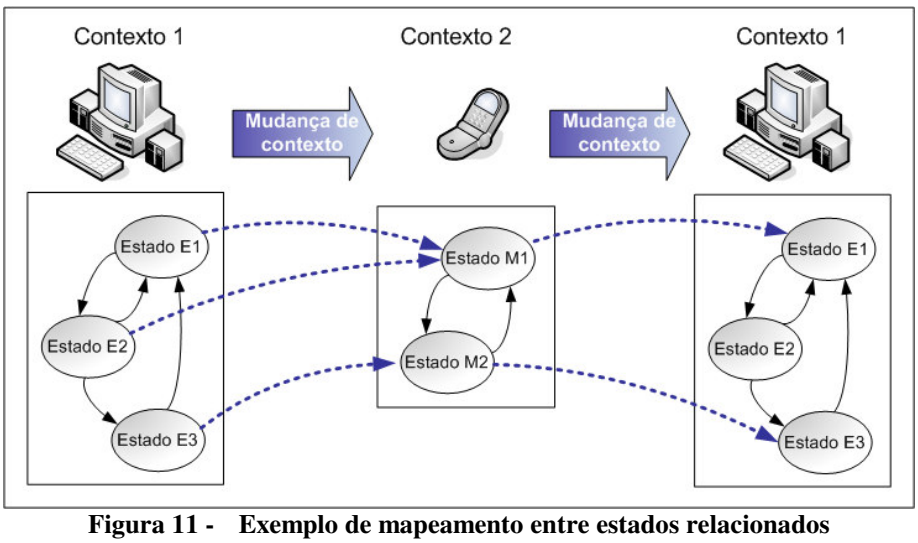

Figura 11 - Exemplo de mapeamento entre estados relacionados a diferentes contextos.

Devido ao seu caráter genérico, independente do jogo, o serviço deve ser informado pelo jogo como este procedimento deve ser realizado. O jogo deve, pois, associar cada estado a um certo número de outros estados, para cada contexto diferente do original.

- **Expiração de ações**

Outro mecanismo oferecido pelo serviço é a utilização de um tempo de expiração para as ações enviada ao jogador. Isto ameniza problemas de inconsistência que podem ocorrer quando um jogador móvel recebe um conjunto de ações e só executa a ação selecionada algum tempo depois. Neste ínterim, o conteúdo da área de interesse pode ter mudado de forma que tal ação não possa mais ser realizada.

Assim, o serviço verifica se o prazo de realização da ação requisitada pelo jogador expirou ou não. Em caso positivo, o jogador deverá receber novamente um conjunto atualizado de ações para escolher qual ação tomar.

4.2.2 Diagrama de seqüência

Com a operação `getActions` (figura 12) o Serviço de Adaptação de Interação obtém as ações de um jogador móvel. Esta operação é chamada pela simulação do jogo sempre que o jogador requisita as ações. Ela tem dois parâmetros: a área de interesse do jogador e o seu personagem.

Assim que a operação é chamada, o serviço chama a operação de mesmo nome do `ActionManager`, que é a classe responsável por armazenar as ações para cada estado dos avatares e também realizar a lógica de obtenção dos estados a partir do conteúdo da área de interesse. Em seguida, é criado um conjunto vazio de ações ao qual serão adicionadas aquelas que satisfizerem determinadas restrições. O `ActionManager` obtém, então, as ações associadas ao estado do avatar para analisar quais delas devem ser enviadas ao jogador.

Para cada uma destas ações, é verificado se ela tem relação com a área de interesse. Em caso negativo, a ação é imediatamente adicionada ao conjunto resultante de ações, visto que nenhuma verificação adicional a respeito dela precisa ser feita. Este é o caso, como já mencionado, de uma operação de se desconectar do jogo, por exemplo.

Contudo, caso a ação se relacione com a área de interesse, deve-se verificar se a área de interesse possui os elementos necessários para que a ação possa ser realizada. Isto é feito verificando-se a relação de cada objeto da área de interesse com aquela ação. Caso o objeto

não tenha relação com a ação, é ignorado. Caso ele tenha relação com a ação, será incluído como uma opção daquela ação. Se a ação tiver ao menos uma opção, isto quer dizer que ela poderá ser realizada naquela área de interesse, sendo incluída no conjunto resultante de ações. Caso contrário, ela será descartada no momento e não será enviada ao jogador.

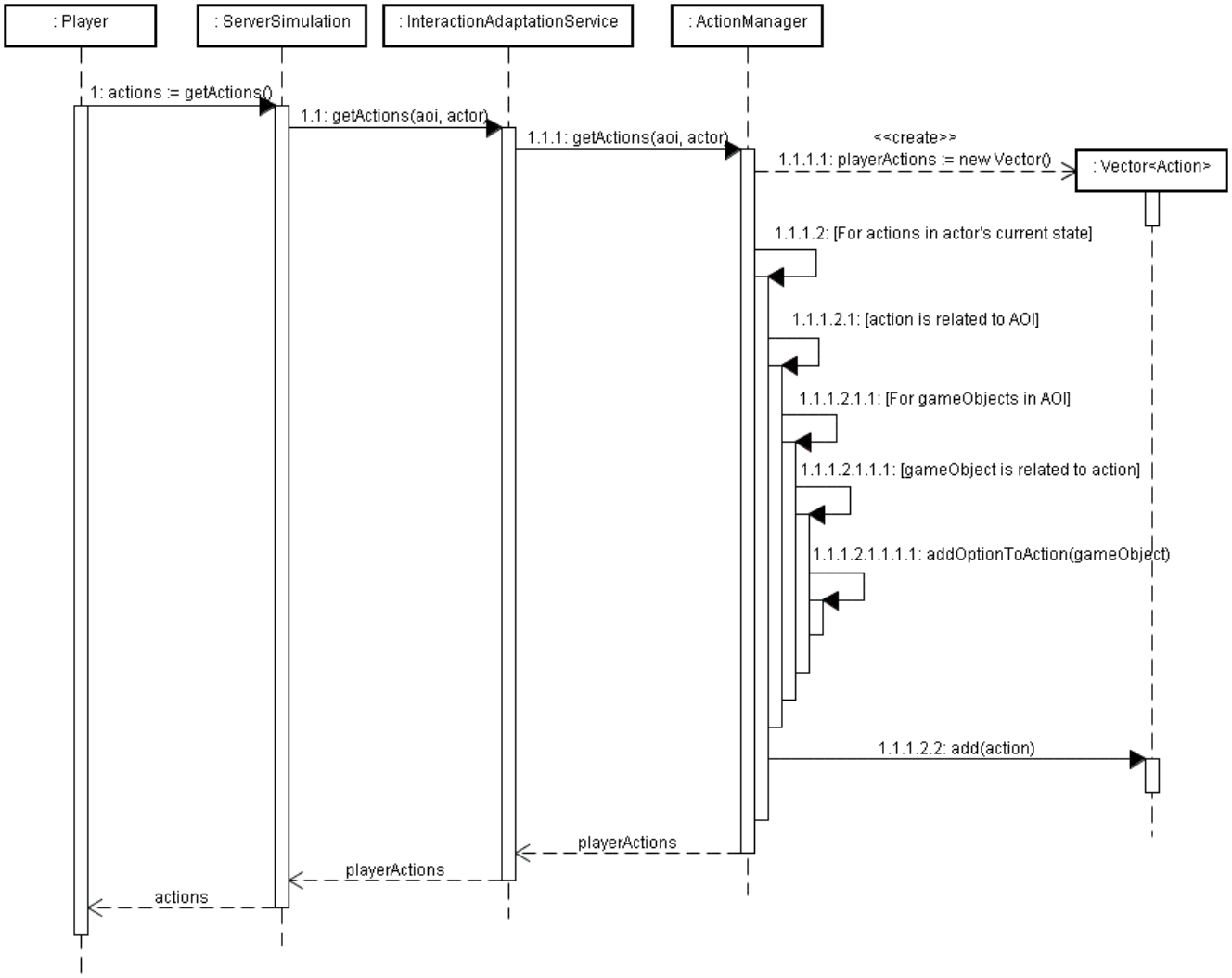

Figura 12 - Diagrama de seqüência da operação getActions do Serviço de Adaptação de Interação.

A operação `decodeAction` (figura 13) do Serviço de Adaptação de Interação é responsável por mapear uma ação do jogador móvel em ações padrões do jogo. Esta operação resulta em um objeto `GameEvent`, que é o evento padrão de comunicação entre o servidor do jogo e clientes estacionários.

Seus parâmetros são informações do jogador, obtidas pela simulação do jogo quando a ação é enviada, e os dados serializados da ação. Tais dados indicam a ação escolhida e qual opção foi selecionada, caso tenha havido alguma opção para aquela ação. Assim que a operação é chamada pela simulação do jogo, o serviço chama a operação `decodeAction` de `ActionManager` para decodificar a ação e obter o evento de jogo relacionado.

O `ActionManager`, por sua vez, apenas cria um novo `GameEventDefault` – implementação padrão da classe abstrata `GameEvent` – com os dados do jogador, da ação e da opção selecionadas pelo jogador. Este evento é então retornado à simulação de jogo, que o processa normalmente como um evento de jogo qualquer.

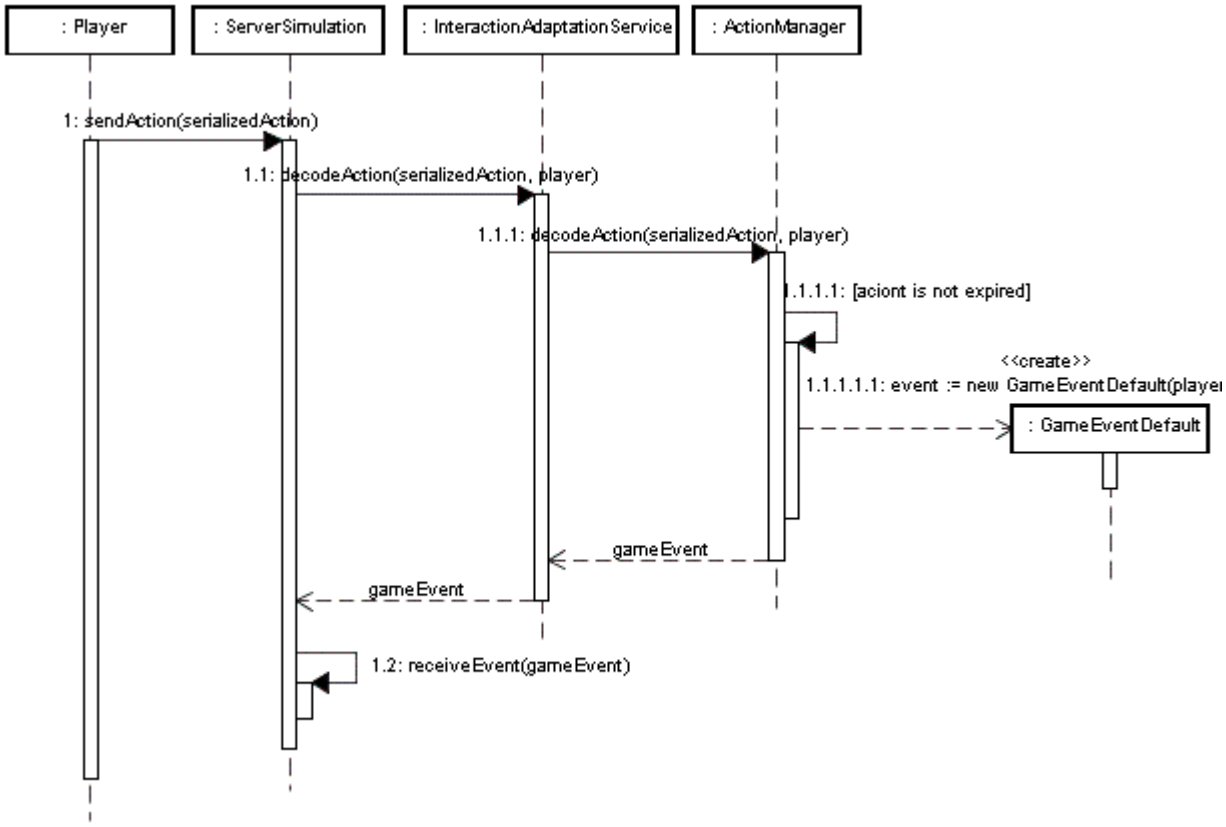

Figura 13 - Diagrama de seqüência da operação decodeAction do Serviço de Adaptação de Interação.

Por meio da operação `getNewState` (figura 14) do Serviço de Adaptação de Interação, a simulação do jogo é capaz de obter o novo estado do avatar de um jogador quando ele muda de contexto de execução. Para tanto, precisa informar o estado atual do avatar e o novo contexto assumido pelo jogador.

Logo que a operação é chamada pela simulação do jogo, é delegada a lógica da operação para um objeto `StatesLevelMapper`, que possui todos os mapeamentos entre cada estado e os estados de outro contexto que lhe são correspondentes. É obtido, então, o objeto `StateLevelMapping` relativo ao estado atual, que contém todos os estados correspondentes a este estado. Finalmente, sua operação `getCorrespondingState` permite obter o estado desejado passando como parâmetro o novo contexto do jogador.

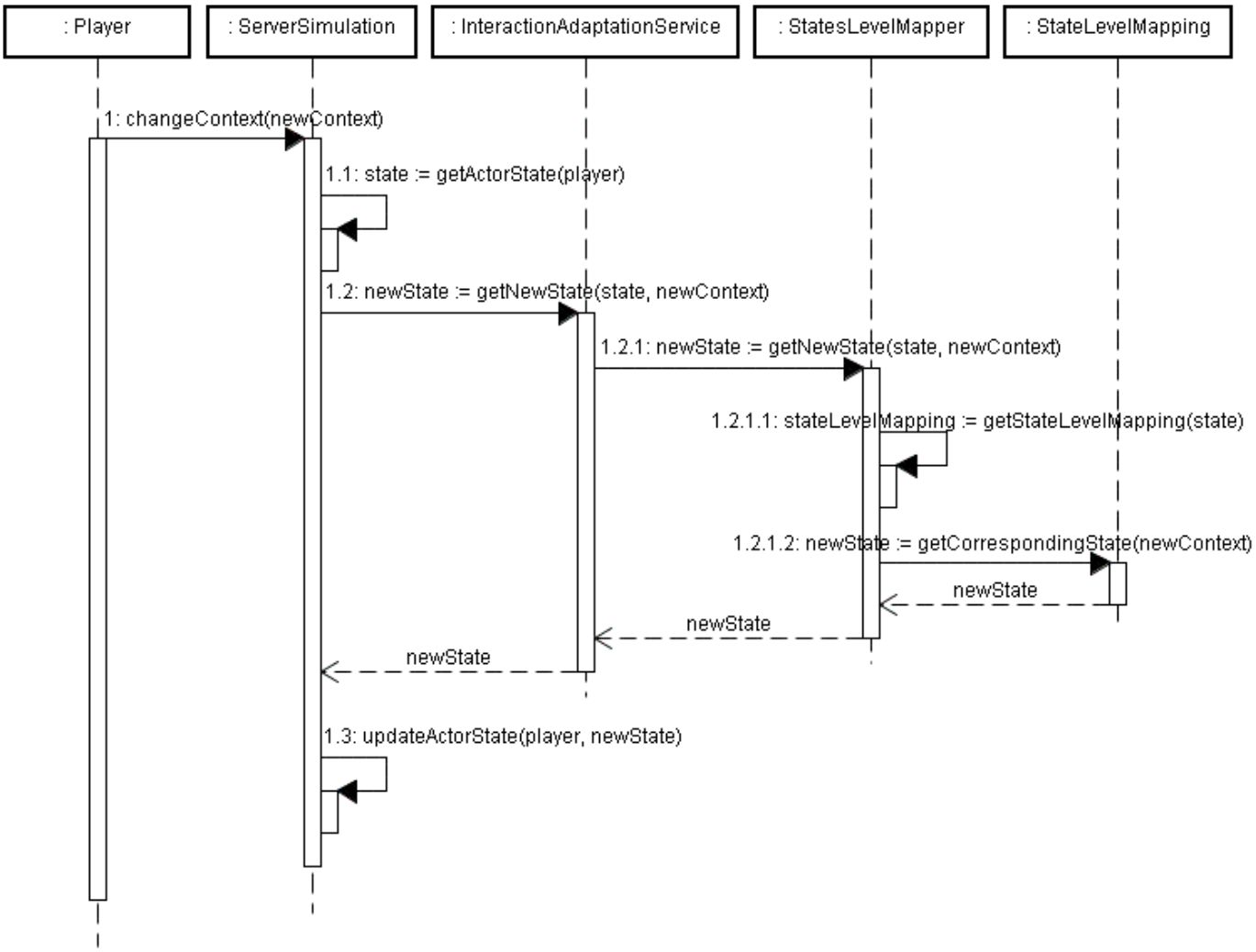

Figura 14 - Diagrama de seqüência da operação getNewState do Serviço de Adaptação de Interação.

4.2.3 Arquitetura proposta

A figura 15 apresenta a arquitetura proposta para o Serviço de Adaptação de Interação como um diagrama de classes em UML.

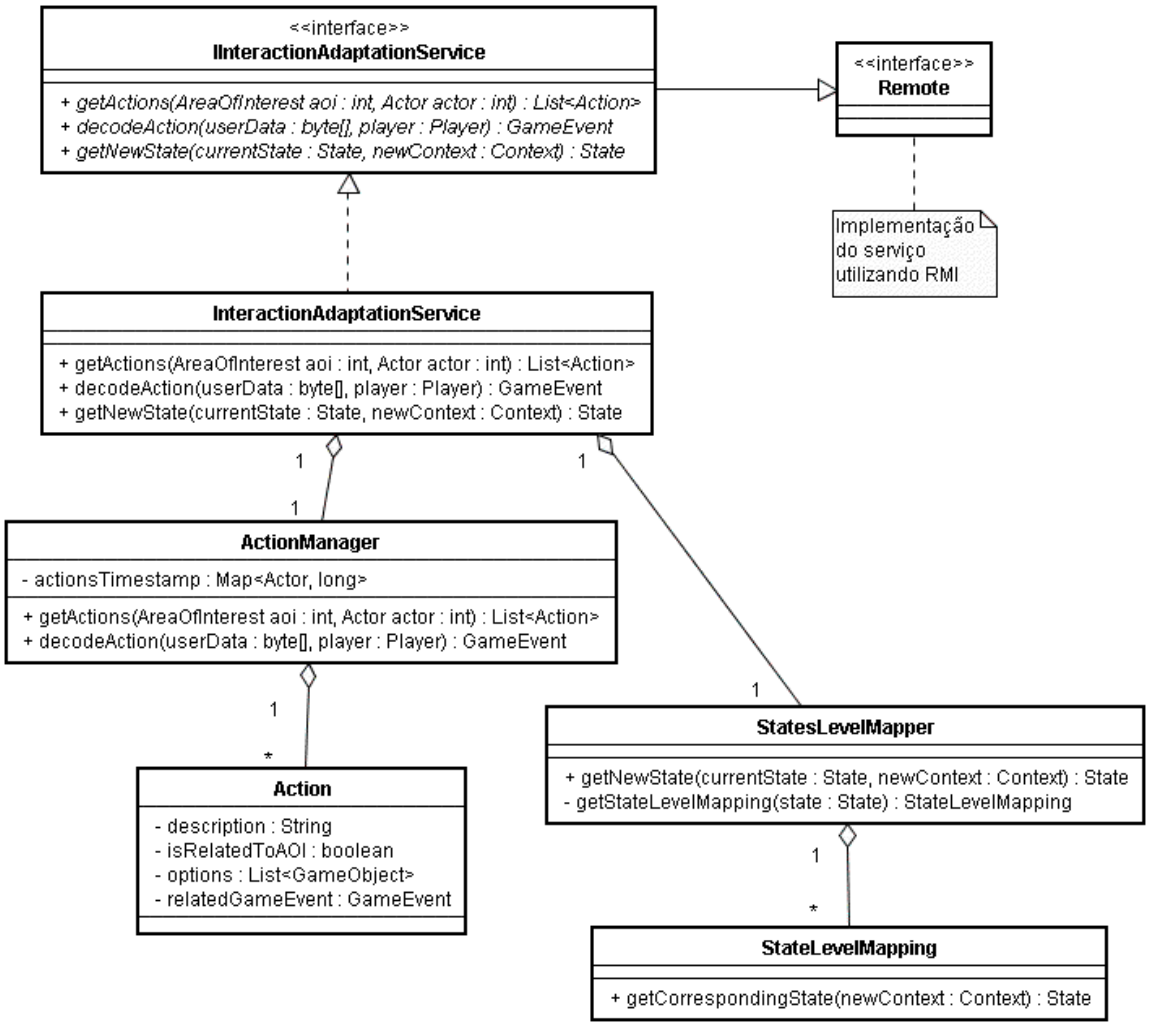

Figura 15 - Diagrama de classes do Serviço de Adaptação de Interação.

4.3 Avaliação dos serviços

Para avaliar os serviços especificados e implementados, pode-se primeiramente analisar se os requisitos não-funcionais descritos no começo deste capítulo (facilidade de uso, flexibilidade e extensibilidade) foram satisfeitos. Embora seja possível argumentar que certas características dos serviços podem – pelo menos parcialmente – satisfazer os requisitos,

reconhece-se que para uma validação formal seria necessária a utilização dos serviços por outras pessoas, que não aqueles envolvidos no desenvolvimento do projeto. Isto, contudo, não foi possível de ser realizado.

Quanto à avaliação da funcionalidade dos serviços, realizada a partir de alguns cenários descritos no capítulo seguinte, consistiu apenas de testes preliminares, sem a presença de usuários. Os testes foram conduzidos em computadores equipados com processadores *Intel Pentium 4* (3.06GHz) e 1GB de memória RAM, executando o sistema operacional *Windows XP*. Os resultados destes testes podem ser verificados no próximo capítulo.

Quanto ao desempenho, a validação deu-se pela constatação de que todos os cenários foram executados sem problemas de desempenho perceptíveis ao jogador. Vale registrar que ao longo do desenvolvimento dos serviços uma série de otimizações foi realizada neste sentido, embora não tenham sido mencionadas por questão de simplificação.

CAPÍTULO 5

Cenários de jogos implementados

Ao longo da implementação dos dois serviços tratados no capítulo anterior, foram criados cenários de jogos para os validar. Tendo sido implementados tanto para computadores pessoais, para dispositivos móveis com interface gráfica e telefones celulares de interface textual, garantem a correção e eficiência das operações realizadas por ambos os serviços em diferentes contextos de execução.

Os cenários foram criados considerando diferentes níveis de imersão e abstração para cada dispositivo utilizado, conceitos já utilizados no *middleware*. Deste modo, a implementação para computadores pessoais exige uma participação mais intensa do jogador e permite a este ter um maior controle do seu avatar, enquanto as implementações para dispositivos móveis visam uma interação de mais alto nível e uma menor quantidade de informações passadas ao jogador.

Estes cenários baseiam-se em situações típicas de um MMORPG e incluem interações do jogador com avatares de outros jogadores e com itens no mundo do jogo. São eles:

- **Obter estado do jogo:** informa ao jogador o conteúdo de sua área de interesse, de diferentes maneiras de acordo com o seu contexto.
- **Movimentação do avatar:** permite ao jogador mover seu avatar livremente pelo mundo virtual, dentro de sua área de interesse e também entre diferentes áreas de interesse.
- **Atacar adversário:** este cenário é um exemplo de interação entre dois jogadores, na qual seus dois avatares lutam entre si.
- **Coletar item virtual:** o avatar coleta um item virtual e o torna indisponível aos demais, modificando, assim, o conteúdo de sua área de interesse.

Nesta seção será detalhada a implementação dos cenários, também para se ter uma idéia do que o desenvolvedor deve realizar com relação às entidades do jogo e à configuração dos serviços. Em seguida, será explicado cada um dos cenários.

5.1 Implementação

Para implementar os cenários foi preciso realizar algumas etapas, também necessárias à criação de qualquer jogo a ser utilizado sobre o *middleware* PM³G. Consistem em criar as entidades do jogo a partir das entidades abstratas já definidas no *framework*, bem como configurar os serviços na inicialização do jogo, para serem utilizados ao longo da simulação.

5.1.1 Definição de entidades do jogo

Várias classes concretas foram criadas, definidas a partir das seguintes classes abstratas do *middleware*:

- **`Player:`** foi criada a entidade `PlayerDefault`, que basicamente acrescenta à classe Player um ator correspondente ao jogador.
- **`GameObject:`** várias subclasses foram criadas, como mostra a figura 16. `MoveableGameObject` refere-se aos objetos que podem ser carregador pelos avatares, e que podem ser `Armor`, `Portion` ou `Weapon`. `Armor` tem as subclasses `Helmet` e `Shield`, que representam, respectivamente, um elmo e um escudo. `HealingPortion` é a porção que pode ser tomada pelo avatar para melhorar sua saúde. `Knife`, `Sword` e `Spear` representam, respectivamente, faca, espada e lança. Além dos `MoveableGameObject`, há duas subclasses de `Performer`: `Dog` e `Wizard`, que é subclasse também de `Actor`. `Dog` é apenas um NPC com o qual o jogador não interage, enquanto `Wizard` é o objeto que representa o avatar do jogador.
- **`State:`** para o contexto estacionário foram definidos quatro estados: `IdleState`, no qual o personagem não faz nada; `MovingState`, que indica movimentação do avatar; `FightingState`, no qual o avatar está lutando contra um adversário; e `RestingState`, estado em que o personagem recupera sua saúde com o passar do tempo e não pode ser atacado. Para os dois contextos móveis, três estados foram definidos: `RestingMobileState`, no qual o avatar descansa de maneira semelhante

ao `RestingState` e também não pode ser atacado; `FightingMobileState`, estado assumido pelo personagem após o jogador enviar o comando de atacar um adversário, e `CollectingMobileState`, estado assumido pelo avatar após o jogador enviar o comando de coletar um item.

- **`Action:`** duas ações foram definidas para os jogadores móveis: `GetItemAction` e `GoFightAction`. A primeira é realizada quando o jogador deseja coletar algum item na seua área de interesse, e a segunda ação corresponde ao comando do jogador para atacar algum adversário. A cada uma delas foi associado o código do evento de jogo correspondente.

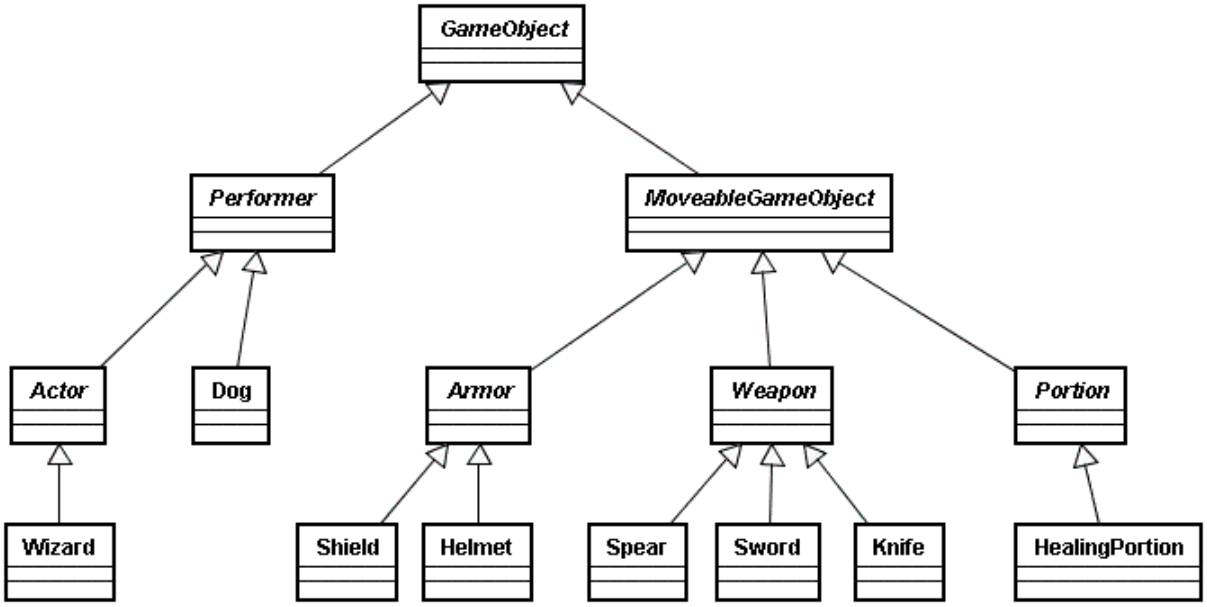

Figura 16 - Diagrama de classes das subclasses de GameObject.

Além destas classes, foram criadas classes concretas de infra-estrutura necessárias para a execução do jogo no servidor e no cliente, e também classes referentes a outros serviços. Seu detalhamento pode ser encontrado em [Tri06b] [Tri06c].

5.1.2 Configuração dos serviços

Embora concebidos para serem o mais genérico possível, tanto o Serviço de Adaptação de Conteúdo como o Serviço de Adaptação de Interação precisam de informações específicas do jogo para funcionar. Deste modo, ao se criar um jogo é importante fazer com que na sua inicialização os serviços sejam configurados corretamente de acordo com os dados do jogo.

- **Serviço de Adaptação de Conteúdo**

Este serviço precisa ter armazenadas todas as estratégias de adaptação que serão utilizadas no jogo. Afinal, cada jogo pode definir um conjunto próprio de estratégias, baseado nos dispositivos para o qual é projetado. Para os cenários implementados, foram definidas três estratégias de adaptação. A primeira, para PCs, não possui módulos lógicos e o módulo de apresentação utilizado é o padrão (`DefaultPresentationMode`). Já a estratégia para dispositivos móveis com interface gráfica utiliza também o módulo de apresentação padrão, mas utiliza os módulos lógicos de escala bidimensional e filtragem de informações. Finalmente, a estratégia de adaptação para telefones celulares de interface textual utiliza o módulo de apresentação textual e os mesmos módulos lógicos da estratégia anterior.

- **Serviço de Adaptação de Interação**

Quanto a este serviço, é necessário ao jogo fornecer duas informações: uma associação entre estados correspondentes para diferentes níveis de abstração e um mapeamento entre os estados as possíveis ações que o avatar pode realizar a partir deles. A primeira informação é necessária para que o serviço possa determinar qual o novo estado a ser assumido por um avatar cujo jogador mudou de contexto. No caso dos estados criados, foi definido que, quando o jogador muda para um contexto de alto nível de abstração, os estados `IdleState`, `MovingState`, e `RestingState` levam para o estado `RestingMobileState`, e `FightingState` leva para `FightingMobileState`. Se o jogador mudar para um contexto de baixo nível, foi definido que `CollectingMobileState`, `FightingMobileState` e `RestingMobileState` levam para `RestingState`.

Ainda com relação ao Serviço de Adaptação de Interação, foi criada somente uma associação entre o RestingMobileState e as ações GoFightAction e GetItemAction. Isto quer dizer que o jogador móvel só poderá realizar ações caso seu avatar esteja neste estado.

5.1.3 Operações básicas para o jogador

Para permitir a utilização dos cenários, foram criadas algumas operações básicas para o jogador. São elas:

- **Conectar-se e desconectar-se do jogo:** estas operações são acessíveis a partir de uma aplicação para PCs, em cuja tela o jogador deve entrar com o comando de *login* acompanhado do seu nome de usuário e senha.
- **Definir contexto de execução:** realizada a partir da mesma aplicação, deve ser sempre chamada antes que o jogador comece a jogar.

5.2 Descrição dos cenários

5.2.1 Obter estado do jogo

Antes de realizar qualquer interação com o seu avatar, o jogador deve estar ciente do conteúdo da sua área de interesse. Este cenário, portanto, fornece informações ao jogador a respeito de sua área de interesse, em uma freqüência, quantidade e nível de detalhamento que depende do seu contexto.

Jogadores estacionários são atualizados constantemente com estas informações, mesmo que não estejam interagindo com o jogo no momento, sendo elas continuamente exibidas. Jogadores móveis, por sua vez, têm que requisitar o estado do jogo sempre que desejarem informações da sua área de interesse.

A quantidade de informações recebidas varia para cada contexto de execução dos jogadores. Os estacionários recebem todas as informações possíveis, sejam elas a respeito dos personagens dos outros jogadores, dos NPCs ou dos itens virtuais. Jogadores móveis, por sua vez não recebem informações acerca dos NPCs. Além disso, para cada contexto móvel o nível de detalhamento das informações é diferente: jogadores que utilizam dispositivos móveis com interface gráfica têm uma noção da posição de cada personagem na área de interesse, enquanto jogadores com aparelhos celulares de interface textual não têm este tipo de informação.

A figura 17 mostra as visões de jogadores em cada um dos contextos para um mesmo instante do jogo. A primeira imagem trata-se da tela da aplicação para PCs, em que são exibidos os avatares, itens virtuais e NPCs. A imagem de baixo e à esquerda corresponde à tela da aplicação para dispositivos móveis com interface gráfica, e exibe apenas avatares e itens virtuais. A menor imagem é uma tela para aparelhos celulares com interface textual, que apenas quantos são os atores e itens presentes na área de interesse.

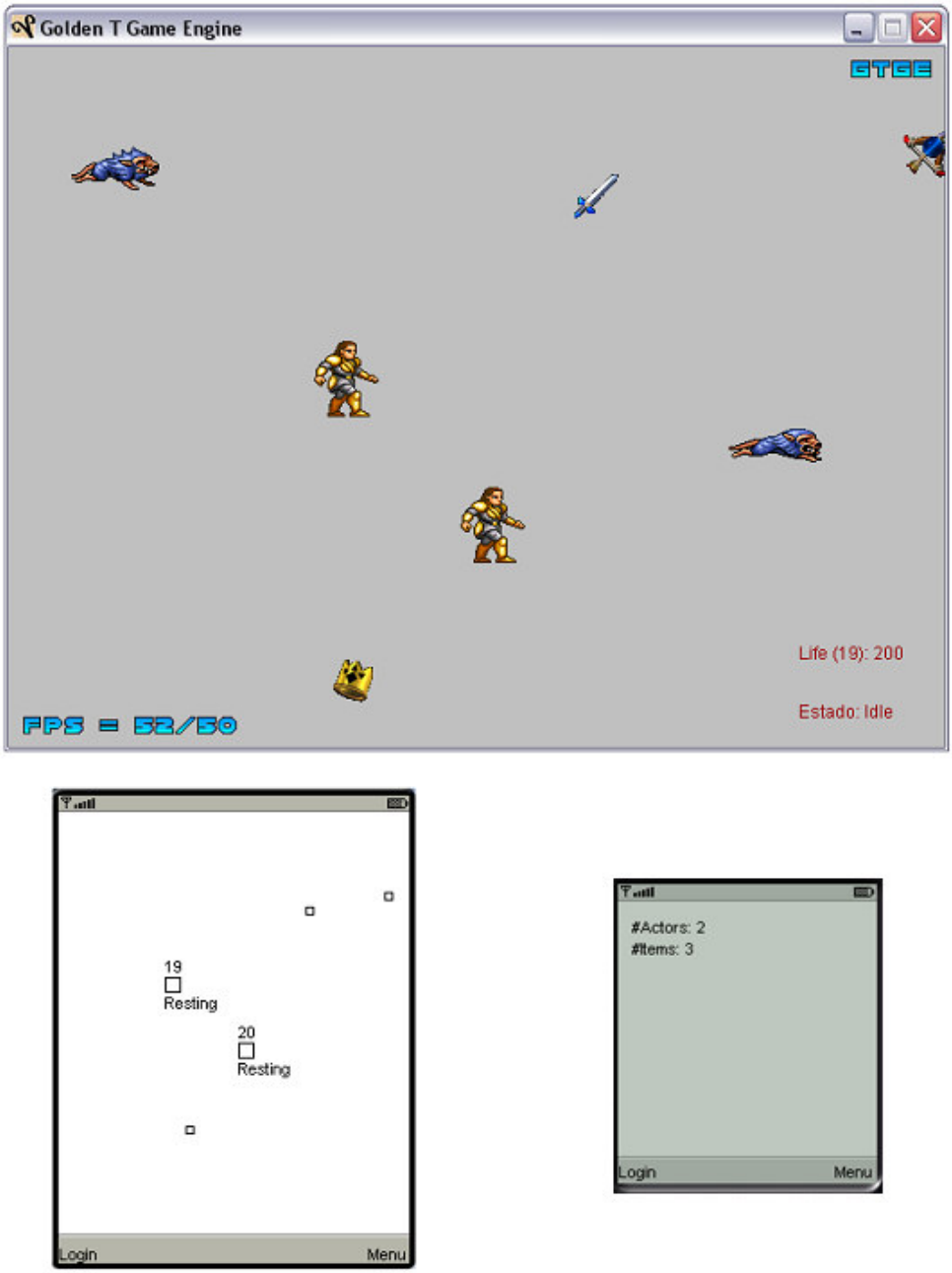

Figura 17 - Diferentes visões do mesmo jogo para cada contexto de execução.

5.2.2 Movimentação do avatar

Neste cenário, o jogador controla livremente a movimentação do seu avatar. Isto pode ser utilizado para um objetivo específico, como um ataque ou coleta de item virtual, mas pode também ser realizado apenas com o propósito de locomover o personagem.

A simples movimentação do avatar, sem fins específicos, foi definida como uma interação de baixo nível, pois exige do jogador um controle minucioso da posição atual do seu personagem. Desta forma, foi implementado apenas para o contexto de computadores pessoais. O jogador interage utilizando o *mouse*, clicando na posição do mundo virtual para a qual deseja que seu personagem se dirija.

5.2.3 Atacar adversário

Para o jogador estacionário, o ataque é realizado quando o personagem se movimenta em direção a outro personagem e, então, colide com o mesmo. Neste momento, os dois personagens entram em estado de luta. Não existe, portanto, um comando explícito para atacar; ao invés disso, o jogador tem que controlar seu personagem e guiá-lo até o adversário.

Por sua vez, o jogador móvel não tem um controle direto da movimentação do seu personagem. Para que seu avatar realize um ataque, ele realiza uma interação de alto nível com o jogo. Caso haja algum adversário na sua área de interesse, será possível a ele realizar o comando de ataque especificando o adversário. Seu avatar, então, se locomoverá de maneira automática até o adversário e os dois começarão a lutar entre si.

A figura 18 exibe um comando explícito para a ação de ataque, disponível para o jogador no contexto móvel. São associados a esta ação os objetos da área de interesse que se relacionam com ela. No exemplo, o personagem do jogador *cagf*, possível alvo da ação, é exibido para que o jogador possa selecioná-lo.

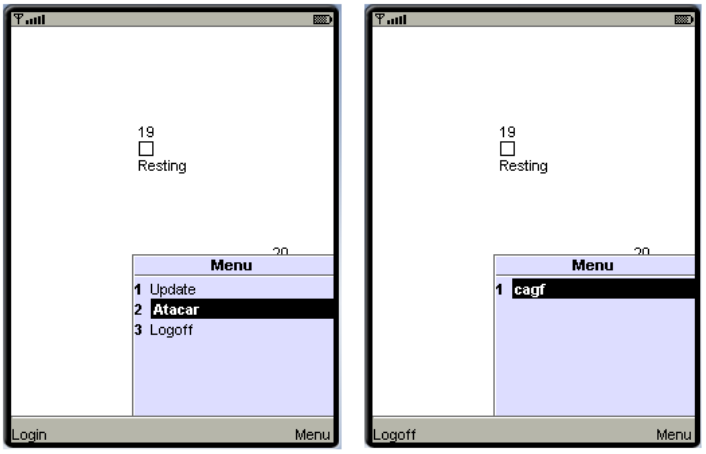

Figura 18 - Comando explícito para atacar o personagem de outro jogador.

5.2.4 Coletar item virtual

De forma semelhante ao ataque a um adversário, a coleta de item virtual é realizada pelo jogador estacionário quando o personagem se movimenta em direção a um item na sua área de interesse e colide com ele. Também não existe um comando explícito para coletar um item virtual, sendo necessário ao jogador dirigir seu personagem até o item desejado.

Já o jogador móvel dispõe de um comando direto, caso haja itens virtuais visíveis na sua área de interesse. Quando a ação é enviada ao servidor, a simulação do jogo movimenta o personagem até o item virtual, até que haja uma colisão e o item seja coletado.

CAPÍTULO 6

Conclusões

Atualmente, a evolução dos dispositivos móveis e da infra-estrutura de redes de computadores e de telefonia móvel aproxima os conceitos de computação pervasiva mais ao cotidiano das pessoas. Jogos que venham a explorar este novo cenário serão inovadores em vários aspectos. Apesar de já existirem iniciativas neste sentido, elas não exploram todo o potencial existente. O *middleware* PM³G, ainda em desenvolvimento, procura, portanto, minimizar esta lacuna na área de entretenimento móvel, auxiliando o desenvolvimento e a execução de jogos multiplataforma.

Os serviços de Adaptação de Conteúdo e de Adaptação de Interação são fundamentais para que os objetivos da plataforma PM³G sejam alcançados. Utilizando a idéia de ciência de contexto (*context-awareness*), estes serviços enriquecem a experiência de jogo dos usuários que utilizem diferentes dispositivos para acessar um mesmo jogo.

Este trabalho traz como principal contribuição o detalhamento da especificação e implementação destes dois serviços. A utilização de diagramas de seqüência e diagramas de classes permite uma compreensão tanto da arquitetura como do funcionamento de ambos os serviços. Conceitos básicos também foram criados, que dão ao desenvolvedor do jogo multiplataforma uma visão mais concreta dos diversos aspectos que o jogo deve abordar.

Alguns cenários de jogos utilizando os serviços também foram concebidos e tiveram sua implementação detalhada neste documento. Foram, assim, descritos alguns passos fundamentais à criação de um jogo multiplataforma que utilize o *middleware* PM³G, servindo de *guideline* para os desenvolvedores destes jogos. Finalmente, pôde-se demonstrar que jogos que utilizem tais cenários – ou cenários semelhantes – podem ser criados utilizando-se a plataforma.

6.1 Trabalhos futuros

Por consistir de um esforço inicial na modelagem e implementação dos serviços abordados, este trabalho não se aprofundou em aspectos técnicos ainda relevantes a jogos multiusuário. Métricas de desempenho e escalabilidade devem ser definidas, e testes precisos ainda são necessários para garantir que os serviços possam ser utilizados em ambientes reais de execução.

Referências

[80105] IEEE 801. IEEE 802.11, The Working Group Setting the Standards for Wireless LANs, maio 2005. http://grouper.ieee.org/groups/802/11/.

[Act01] Activision. Return to Castle Wolfenstein. Disponível em http://games.activision.com/games/wolfenstein/. Acesso em setembro de 2006.

[Ale02] Thor Alexander. *A Flexible Simulation Architecture for Massively Multiplayer Games*. In Game Programming Gems 3, pages 506 - 519. Charles River Media, 2002.

[Bat06] Battle.net. Battle.net, 2006. Disponível em http://www.battle.net/. Acesso em setembro de 2006.

[BBV03] David Brackeen, Bret Barker, Laurence Vanhelsuwé. *Developing Games in Java*. New Riders Publishing, 2003.

[BE06a] Blizzard Entertainment. Diablo, 2006. Disponível em http://www.blizzard.com/diablo/. Acesso em setembro de 2006.

[BE06b] Blizzard Entertainment. Starcraft, 2006. Disponível em http://www.blizzard.com/starcraft/. Acesso em setembro de 2006.

[BE06c] Blizzard Entertainment. The World of Warcraft Community Site, 2006. Blizzard, Inc. Disponível em http://www.worldofwarcraft.com/. Acesso em setembro de 2006.

[Ber96] Philip A. Bernstein. *Middleware: a model for distributed system services*. *Commun*. ACM, 39(2):86–98, 1996.

[Blu03] Bluetooth.org. The Official Bluetooth Membership Site, 2003. Disponível em http://www.bluetooth.org/spec/. Acesso em setembro de 2006.

[CDK05a] George Couloris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concept and Design*, chapter Mobile and Ubiquitous Computing, pages 657 - 719. Addison Wesley, 4th edition, June 2005.

[CDK05b] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*. Addison Wesley/Pearson Education, 4th edition, June 2005.

[Cec05] Fábio Reis Cecin. *FreeMMG: uma arquitetura cliente-servidor e par-a-par de suporte a jogos maciçamente distribuídos*. Master's thesis, Instituto de Informática. Universidade Federal do Rio Grande do Sul, Janeiro 2005.

[CFG+03] Adrian David Cheok, Siew Wan Fong, Kok Hwee Goh, Xubo Yang, Wei Liu, and Farzam Farzbiz. *Human pacman: a sensing-based mobile entertainment system with ubiquitous computing and tangible interaction*. In NETGAMES '03: Proceedings of the 2nd workshop on Network and system support for games, pages 106–117. ACM Press, 2003.

[CPP00] The C++ Resources Network. cpluspluc.com. Disponível em http://www.cplusplus.com/. Acesso em setembro de 2006.

[Day06] Daydream. Bot Fighters. Disponível em http://www.botfighters.com. Acesso em setembro de 2006.

[Dir06] Microsoft Corporation. Microsoft DirectX: Home Page. Disponível em http://www.microsoft.com/directx/. Acesso em setembro de 2006.

[Emm00] Wolfgang Emmerich. *Software engineering and middleware: a roadmap*. In Proceedings of the conference on The future of Software engineering, pages 117–129. ACM Press, 2000.

[Far05] Fairfield, J., 2005, 'Virtual Property', *Boston University Law Review*, vol. 85, no.4, pp.1047-1102.

[Fox03] David Fox. *Massively Multiplayer Game Development (Game Development)*, volume 1, chapter Small Portals: Tapping into MMP Worlds via Wireless Devices, pages 244 – 254. Charles River Media, Inc., Rockland, MA, USA, 2003.

[FRC03] Cláudio Fernando Resin Geyer Fábio Reis Cecin, Jorge Luis Victória Barbosa. *Freemmg: An hybrid peer-to-peer, client-server and distributed massively multiplayer game simulation mode*. In Proceedings of the 2th Brazilian Workshop on Games and Digital Entertainment, November 2003.

[GC06] Gravity Corp. Ragnarok Online. Disponível em http://www.ragnarokonline.com/. Acesso em setembro de 2006.

[GPS06] National Space-Based Positioning, Navigation, and Timing Coordination Office. Global Positioning System. Disponível em http://www.gps.gov/. Acesso em setembro de 2006.

[GSM06] GSM Association. GSM World – GPRS Platform. Disponível em http://www.gsmworld.com/technology/gprs/index.shtml. Acesso em setembro de 2006.

[Hel03] Ville Helin. *Development of modern multiplayer games*. Master's thesis, Helsinki University of Technology, July 2003.

[HIW04] JungHyun Han, Hoh Peter In, and Jong-Sik Woo. *Towards situation-aware cross-platform ubi-game development*. In APSEC, pages 734–735, 2004.

[IBM99] IBM. Microsoft DirectX: Home page, 1999.

[IDG02] IGDA. IGDA online games white paper, 2002. Disponível em http://www.igda.org/online/IGDAOnlineGamesWhite_paper_2002.pdf. Acesso em setembro de 2006.

[Ign06] IGN. Online Game Market to Reach $13 Billion. Disponível em http://xbox.ign.com/articles/711/711480p1.html. Acesso em agosto de 2006.

[IPe06] Integrated Project on Pervasive Gaming. Disponível em http://www.iperg.org. Acesso em setembro de 2006.

[iS06] id Software. id Software, 2006. Disponível em http://www.idsoftware.com/. Acesso em setembro de 2006.

[KLXH04] Björn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. *Peer-to-peer support for massively multiplayer games*. In Proceedings of the 23rd

Conference of the IEEE Communications Society. IEEE Computer Society, July 2004.

[Kri03] Jan Krikke. *Samurai romanesque, j2me, and the battle for mobile cyberspace*. IEEE Computer Graphics and Applications, 23(1):16–23, 2003.

[KW05] Elina M.I. Koivisto and Christian Wenninger. *Enhancing player experience in mmorpgs with mobile features*. In 2nd International Digital Games Research Association Conference, 2005.

[MC05] M1nd Coorporation. Alien Revolt, Maio 2005. Disponível em http://www.alienrevolt.com/pt. Acesso em setembro de 2006.

[MG05] Meantime Games. Meu Big Brother, January 2005. Disponível em http://bbb.globo.com/BBB5/0,24118,BCW2-4057-20,00.html. Acesso em outubro de 2006.

[Mic04] Microsoft. Microsoft game studios | the official age of empires website, 2004. Disponível em http://www.microsoft.com/games/empires/. Acesso em setembro de 2006.

[MWO06] Merrian-Webster Online. Disponível em http://www.m-w.com. Acesso em julho de 2006.

[NCs06a] NCsoft. Lineage 2, The Chaotic Chronicle – the official web site, 2005. NCsoft, Inc. Disponível em http://www.lineage2.com. Acesso em setembro de 2006.

[NCs06b] NCsoft. Lineage, Dark Conquest – the official web site, 2005. NCsoft, Inc. Disponível em http://www.lineage.com. Acesso em setembro de 2006.

[NG04] Nokia N-Gage. Nokia N-Gage | home, 2004. Disponível em http://www.ngage.com. Acesso em setembro de 2006.

[NG06] Newt Games. Mogi item hunt. Disponível em http://www.mogimogi.com. Acesso em setembro de 2006.

[Nin05] Nintendo. Nintendo's official site for nintendo ds, 2005. Disponível em http://www.nintendods.com/. Acesso em setembro de 2006.

[Nok03] Nokia. Tibiame Case Study, June 2003. Disponível em http://ncsp.forum.nokia.com/downloads/nokia/documents/Tibia_v_1_0.pdf. Acesso em setembro de 2006.

[Nok06a] Nokia. Nokia N-Gage | Pocket Kingdom. Disponível em http://arena.ngage.com/n-gage/web/en/pocketkingdom/index.jsp. Acesso em setembro de 2006.

[Nok06b] Nokia. HINTER WARS | Multiplayer Online/Mobile Games. Disponível em http://www.hinterwars.com/main_v2.html. Acesso em setembro de 2006.

[Obj05] ObjectWeb. ObjectWeb - HomePage, 2005. Disponível em http://www.objectweb.org/. Acesso em setembro de 2006.

[OMAOa] Open Mobile Alliance OMA. Oma Technical Section - mobile games interoperability forum. Disponível em http://www.openmobilealliance.org/tech/affiliates/mgif/mgifindex.html. Acesso em setembro de 2006.

[OMAOb] Open Mobile Alliance OMA. Oopen Mobile Alliance - home. Disponível em http://www.openmobilealliance.org. Acesso em setembro de 2006.

[OMG03] OMG. Unified modeling language. Specification v1.5, Object Management Group, March 2003. Disponível em http://www.omg.org/technology/documents/formal/uml.htm. Acesso em setembro de 2006.

[oST00] National Institute of Standards and Technology. Smart spaces and their related areas, 2000.

[PDGSS05] R. Pellerin, F. Delpiano, E. Gressier-Soudan, and M. Simatic. *Gasp : Un intergiciel pour les jeux en réseaux multijoueurs sur téléphones mobiles*. Deuxièmes Journées Francophones: Mobilité et Ubiquité 2005 - UBIMOB'05, pages 61 - 64, 2005.

[PQ06] Planet Quake. Quake, 2005. Disponível em http://www.planetquake.com/quake1/. Acessado em setembro de 2006.

[Pri00] M. Pritchard. *How to hurt the hackers: The scoop on internet cheating and how you can combat it*, July 2000. Disponível em http://www.gamasutra.com/features/20000724/pritchard_01.htm. Acesso em outubro de 2006.

[PW02a] Lothar Pantel and Lars C. Wolf. *On the impact of delay on real-time multiplayer games*. In Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video, pages 23–29. ACM Press, 2002.

[PW02b] Lothar Pantel and Lars C. Wolf. *On the suitability of dead reckoning schemes for games*. In Proceedings of the 1st workshop on Network and system support for games, pages 79–84. ACM Press, 2002.

[QI06] QUALCOMM Incorporated. QUALCOMM BREW Home, 2006. Disponível em http://brew.qualcomm.com/brew/pt/. Acesso em outubro de 2006.

[SAP+05] C. Silva *et al*. *Uma Plataforma para Jogos Móveis Massivamente Multiusuário*. In: SBGames 2005 - WJogos 2005, 2005.

[SE05] Sony Entertainment. Playstation portable - psp, 2005. Disponível em http://www.us.playstation.com/psp.aspx. Acesso em setembro de 2006.

[Seg06] Sega, Inc. The Official Sega website. Disponível em http://www.sega.com. Acesso em setembro de 2006.

[SKH01] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. *Aspects of networking in multiplayer computer games*. In LooWai Sing,Wan Hak Man, andWongWai, editors, Proceedings of International Conference on Application and Development of Computer Games in the 21st Century, pages 74–81, Hong Kong SAR, China, November 2001.

[SM06a] Sun Microsystems. Java Platform, Micro Edition, 2006. Disponível em http://java.sun.com/javame/. Acesso em julho de 2006.

[SM06b] Sun Microsystems. Java Platform, Standard Edition, 2006. Disponível em http://java.sun.com/javase/. Acesso em julho de 2006.

[SMS99] SMS Forum. SMS Forum. Disponível em http://smsforum.net/. Acesso em setembro de 2006.

[SO05] Symbiam One. Symbiam Mobile Operational System, January 2005. Disponível em http://www.symbianone.com. Acesso em setembro de 2006.

[Stu04] Microsoft Game Studios. Welcome to Mythica, 2004. Disponível em http://mythica.com. Acesso em setembro de 2006.

[TdAdAL+03] Dante Gama Torres, Gustavo Danzi de Andrade, André Roberto Gouveia do Amaral Leitão, Igor de Andrade Lima Gatis, Pedro Henrique de Macêdo, and Geber Lisboa Ramalho. *Uma api para a criação de jogos multi-usuário de turno em telefones móveis*. In Proceedings of the 2th Brazilian Workshop on Games and Digital Entertainment, November 2003.

[Ter02] Terazona: Zona application framework whitepaper, 2002. Zona Inc. Disponível em http://www.zona.net/whitepaper/Zonawhitepaper.pdf. Acesso em agosto de 2006.

[Tri06a] Fernando Trinta. *Suporte a Pervasividade em Jogos Massivamente Multiusuário*. Monografia de Qualificação & Proposta de Tese de Doutorado, Centro de Informática, Universidade Federal de Pernambuco, 2006.

[Tri06b] Fernando Trinta. *Middleware Services for Pervasive Multiplatform Networked Games*. 5Th Workshop on Network and System Support for Games - Netgames'06. 30-31 October 2006. Singapore.

[Tri06c] Fernando Trinta. *Uma Proposta de Cenários e Serviços de Suporte para Jogos Multiusuário Multiplataforma Pervasivos*. WebMedia 2006 - Brazilian Symposium on Multimedia and the Web. 19-22 November 2006, Natal, Brazil.

[Ubi04] Ubisoft. Alexander, 2004. Disponível em http://alexander-thegame.com/us/. Acesso em setembro de 2006.

[UMT] UMTS Forum. UMTS Forum : UMTS Forum Home. Disponível em http://www.umts-forum.org/. Acesso em setembro de 2006.

[Wal04] Gordon Walton. *Nine things to look for in the next generation of mmog*, 2004. Disponível em http://www.gdconf.com/conference/archives/2004/walton_gordon.ppt. Acesso em setembro de 2006.

[WC03] Leo Sang-Min Whang, Geun-Young Chang *Lifestyles of virtual world residents, living in the on-line game, "Lineage"*. In: Cyberworlds, 2003. Proceedings. 2003 International Conference, pages 18 – 25, 2003.

[Wik06] Wikipedia. *History of computer and video games*, 2005. Disponível em http://en.wikipedia.org/wiki/History_of_computer_and_video_games. Acesso em outubro de 2006.

[Yan03] Jeff Yan. *Security design in online games*. In Proceedings of the 19th Annual Computer Security Applications Conference, page 286. IEEE Computer Society, 2003.

[YC02] J. Yan and H-J Choi. *Security issues in online games*. In The Electronic Library: international journal for the application of technology in information environments, volume 20, 2002.

[YDr06a] YDreams. YDreams: Home. Disponível em http://www.ydreams.com/. Acesso em outubro de 2006.

[YDr06b] YDreams. Undercover. Disponível em http://hk.playundercover.com. Acesso em setembro de 2006.