Jogos Multiusuário Distribuídos

Fábio Cecin[1] & Fernando Trinta[2]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

[2]Mestrado em Informática Aplicada – Universidade de Fortaleza

Av. Washington Soares, 1321 – sala J30 – CEP 60811-905 – Fortaleza – CE – Brasil.

fcecin@inf.ufrgs.br, trinta@unifor.br

Abstract. *This tutorial begins with an overview of "multiplayer on-line games" in general, which are games that allow for multiple players to interact with each other through a network. From there, it then conceptualizes three concrete multiplayer gaming scenarios: (i) massive, (ii) mobile and (iii) pervasive, and from those concepts it offers a perspective for a new concept: the (iv) massively multiplayer multiplatform game. After a discussion on the modeling and implementation alternatives for developing an engine for those game scenarios, the tutorial walks through a set of published Java APIs that covers modules such as networking, simulation and physics that are key to implementing a client-server multiplayer engine.*

Resumo. *Este tutorial inicia com uma visão geral de "jogos on-line multijogador" em geral, que são jogos que permitem a múltiplos jogadores interagirem através de uma rede. Após, o tutorial conceitua três cenários concretos para jogos multijogador: (i) massivos, (ii) móveis e (iii) pervasivos, e a partir destes conceitos o texto oferece uma perspectiva para um novo: (iv) jogos multijogador massivos multiplataforma. Após uma discussão sobre as alternativas de modelagem e implementação de um motor para estes cenários, o tutorial disseca as interfaces de programação de um conjunto de bibliotecas Java que cobrem módulos como rede, simulação e física, que são essenciais para a implementação de um motor cliente-servidor para jogos multijogador.*

1. Introdução

Jogos Multiusuário Distribuídos vêm se tornando um tópico de pesquisa em grande evidência. Estas aplicações são essencialmente simulações interativas em tempo-real que permitem que uma pessoa chamada jogador utilize um computador conectado a alguma rede para interagir, em um ambiente virtual, com outros jogadores [Cec05]. Em sua fase inicial, a aplicação de jogos multiusuário distribuídos era limitada a redes locais com poucos usuários, geralmente na ordem de dois a oito jogadores. Com o advento da Internet e melhoria da qualidade das conexões à rede mundial de computadores, jogos multiusuário distribuídos permitiram que jogadores geograficamente espalhados pudessem interagir em uma mesma partida. Segundo dados publicados pelo DFC Intelligence Institute (http://www.dfcint.com/), instituto de pesquisa especializado em

mercado de entretenimento digital, o segmento de jogos *on-line* deverá atingir US$ 13 bilhões ao final do ano de 2011.

O sucesso de jogos multiusuário fundamenta-se em certos fatores, sendo que dois pontos merecem destaque: *(i)* a competição entre humanos mostra-se mais atraente e desafiadora que as mais avançadas técnicas de Inteligência Artificial utilizadas em jogos monousuário [Cen01] e *(ii)* estes promovem aspectos sociais interessantes ao permitir a interação entre participantes, mesmo sem um conhecimento prévio entre os mesmos.

Da implementação do primeiro jogo multiusuário até os dias atuais, os desafios encontrados por desenvolvedores destas aplicações foram inúmeros. Primeiramente, jogos são tradicionalmente impulsionados por inovações tecnológicas. Desta forma, jogos multiusuário vêm se moldando a novas tecnologias, e hoje, sua aplicação vai desde de jogos simples em turnos baseados em macromedia flash, passando por jogos em dispositivos móveis até ambientes virtuais persistentes de larga escala, os chamados Jogos Massivamente Multiusuários – MMG (*Massively Multiplayer Games*). Apesar de possuírem requisitos particulares a cada domínio de aplicação, cada um destes diferentes cenários aponta apresentam desafios comuns relacionados com questões como: abrangência de comunicação, latência, largura de banda, bem como capacidade de processamento dos dispositivos envolvidos.

Este documento apresenta um resumo das questões que envolvem o projeto de um jogo multiusuário. Para isto, são detalhados como aspectos tecnológicos influenciam a implementação de jogos multiusuário em geral, além de apresentar as técnicas mais utilizadas para atenuar o impacto destes problemas. São apresentados conceitos gerais e cenários atuais de utilização de jogos multiusuário. Em seguida são apresentados seus desafios e as soluções clássicas a estes problemas.

2. Contextualização

Jogos multiusuário distribuídos pertencem à categoria de aplicações de espaço compartilhado das quais fazem parte simulações militares, ambientes virtuais distribuídos e trabalho colaborativo apoiado por computador – do inglês, *Computer Supported Collaborative Work (CSCW)*. 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 espaço compartilhado entre jogadores, onde o jogo é realizado.

No entanto, mesmo classificados como aplicações de espaço compartilhado, jogos multiusuário apresentam características peculiares. Primeiramente, seu foco está na competição entre seus usuários, ao contrário das demais aplicações de CSCW, onde os usuários colaboram para a realização de alguma tarefa. Esta característica influencia diretamente um segundo aspecto peculiar a jogos multiusuário. Um sistema de suporte para jogos deve evitar que os jogadores possam alterar o estado do jogo de forma indevida, ou seja, de forma que as regras do jogo sejam burladas. De forma mais sucinta, jogos multisuário devem ser intolerantes à trapaça [Cec05]. Enquanto outros ambientes de espaço compartilhado geralmente utilizam redes privadas, com participantes confiáveis, jogos multiusuário podem utilizar redes públicas, onde certos jogadores fazem uso de meios antiéticos para obtenção de vantagens ilegais em relação a jogadores honestos.

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 contraste, um *jogo baseado em turnos* é um jogo onde o jogador envia comandos de acordo com o tempo da simulação. Neste caso, há uma alternância entre as ações de cada jogador, onde cada participante tem bem definido o seu momento de realizar uma jogada. Por exemplo, em um jogo de xadrez, 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 de jogo são comumente encontrados na literatura: jogos em primeira pessoa e jogos de estratégia em tempo-real – do inglês, *Real-Time Strategy (RTS)*.

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 Wolfstein* (2004) apresentado na Figura 1(a). Esta perspectiva dá ao jogador uma intensa sensação de imersão no jogo, exigindo reflexos rápidos e respostas quase instantâneas.

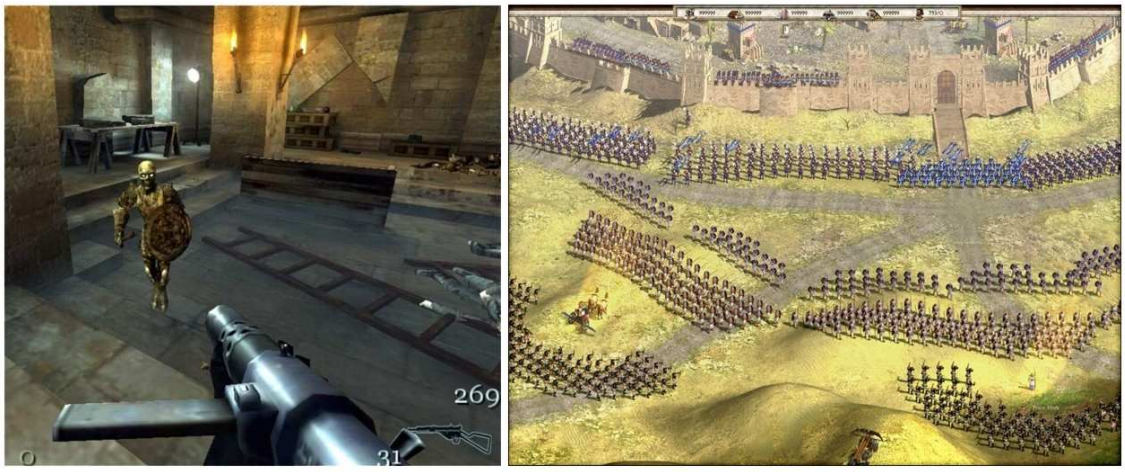


Figura 1 – Exemplos de jogos (a) de Primeira Pessoa e (b) de Estratégia em tempo-real.

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. *Alexander* (2004), apresentado na Figura 1(b), exemplifica um jogo de estratégia em tempo-real.

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 propõe sistemas de suporte para jogos de ação, a preocupação central geralmente é com os comandos que possuem fortes restrições de tempo.

3. Cenários Atuais para Jogos Multiusuário

O aspecto multiusuário em jogos eletrônicos apresenta diferentes cenários para sua aplicação. Estes cenários, não por coincidência, são uma conseqüência da evolução tecnológica vivida desde as primeiras experiências em redes locais até a utilização de dispositivos móveis como plataforma para jogos pervasivos nos dias atuais. Esta seção apresenta os principais cenários para jogos multiusuário em uma linha de evolução tecnológica e temporal.

3.1. 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. Nestes jogos, a máquina de um jogador ou uma máquina dedicada 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 pelos objetivos específicos de cada jogo, como um limite de vinte voltas em uma corrida de carros. Típicas partidas duram alguns minutos ou algumas horas, e o resultado de uma partida não afeta o estado da partida seguinte.

Outra característica destes jogos é seu número de jogadores. Os primeiros jogos multiusuário permitiam de dois a oito jogadores em uma mesma sessão. Estes eram preferencialmente voltados para redes locais, devido ao atraso mínimo necessário para sua jogabilidade. Com a melhoria das tecnologias de comunicação e o aumento do poder de processamento dos computadores pessoais, este número elevou-se, possibilitando até 64 usuários em uma única sessão hospedada na máquina de um jogador. Exemplos destes jogos são *Counter-Strike* [Ste03] e *Starcraft* [BE05].

3.2. Jogos Multiusuário Massivos (MMG)

A popularização do acesso à Internet via banda larga permitiu um aumento considerável na escala de usuários em um jogo multiusuário distribuído, criando os Jogos Massivamente Multiusuário, do inglês, *Massively Multiplayer Games (MMG)*. Estes jogos diferem do cenário anterior por duas características adicionais. Primeiro, a magnitudade do número de jogadores concorrentes é tipicamente na ordem de 10^4 ou

ainda maior [CG04]. Segundo, MMGs dão suporte a uma simulação em constante execução, independente da presença de jogadores, onde o estado do jogo é gravado em algum meio de persistência. Estes jogos, também chamados de mundos contínuos ou persistentes, indicam que ações realizadas por jogadores têm capacidade de moldar permanentemente o mundo virtual. Em outras palavras, um MMG pode continuar sem perspectiva de encerramento.

O modelo de negócios para Massively Multiplayer Games (MMG) difere dos jogos tradicionais. Um MMG é encarado como um serviço oferecido pela empresa mantenedora responsável pela infra-estrutura de suporte ao correto funcionamento do jogo, onde cada usuário é cobrado periodicamente pelo acesso a esta infra-estrutura. Alguns dos mais populares MMGs como *Everquest* [Eve06], *Ultima Online* [Ori02] e *Ragnarok* [Rag05] possuem mais de 200 mil assinantes. Expoentes como os jogos da série *Lineage* [NCs05b] [NCs05a] e o *World of Warcraft* [BE06] ultrapassam milhões de usuários[Woo06], como apresentado na Figura 2.

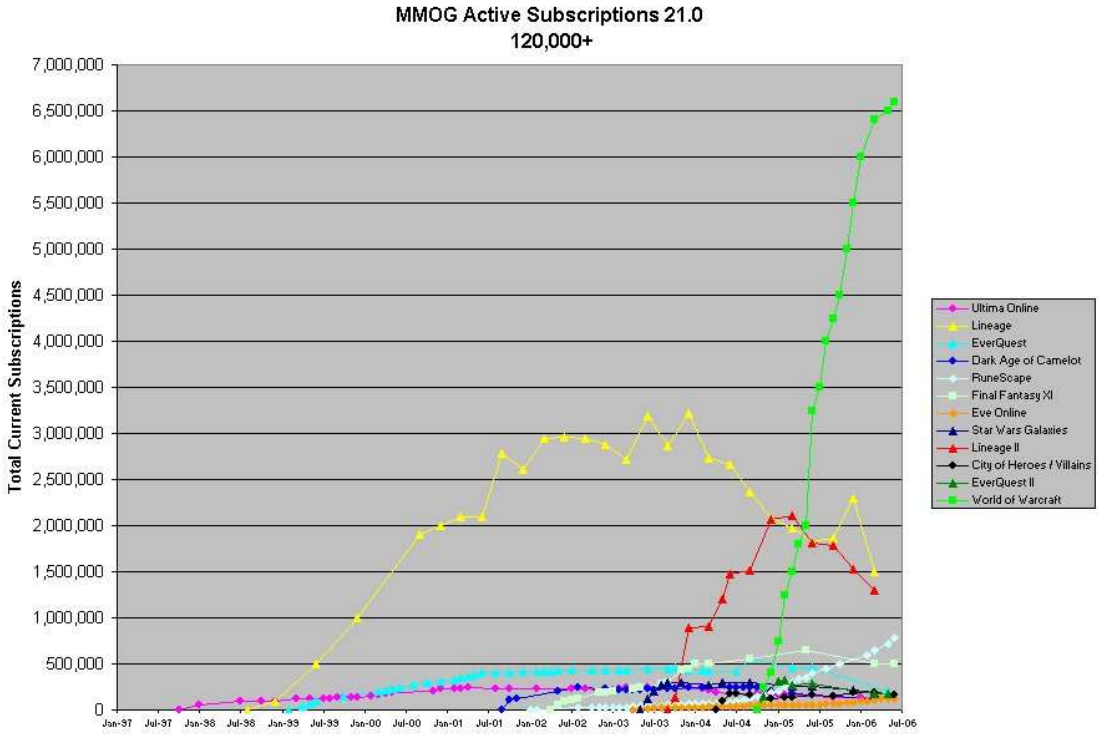


Figura 2 – Crescimento do Número de Assinantes de MMGs atuais.

Porém, apesar de seu sucesso e potencial, MMGs apresentam uma série de problemas ainda a serem solucionados, principalmente no que diz respeito à 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 MMGs, como a própria Microsoft [MGS04].

3.3 Jogos Móveis Multiusuário

A computação móvel é o mais recente cenário de aplicação para jogos multiusuário. Este cenário é conseqüência de diversos fatores. A popularização de dispositivos

móveis, o aumento do poder computacional destes dispositivos e o surgimento de plataformas abertas como J2ME [SM05] 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 *General Packet Radio Service (GPRS)*, *bluetooth* [Blu04] e Wi-Fi (802.11) [IEE05], 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 com o *Nokia N-Gage*™[Nok04], *NintendoDS*™[Nin05] e *Playstation Portátil*™ [SE06] (Figura 3) 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.

Figura 3 – Plataformas de Jogos Móveis Multiusuário

De fato, o uso da tecnologia de comunicação acaba sendo um fator determinante no projeto de um jogo multiusuário móvel. O uso das redes de operadoras de meio físico, sejam elas analógicas, de segunda (2G e 2.5G) ou terceira (3G) geração, utilizam tecnologias que apresentam latências elevadas 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 de meio físico. Por tais motivos, estas redes limitam sua utilização a jogos não tão interativos, como jogos em turno (Seção 2.1), exemplificado pelo jogo *Batalha Naval* [TdAdAL+03].

Uma alternativa para criação de jogos multiusuário mais interativos em dispositivos móveis, mesmo em face aos problemas de conexão de redes sem fio são os chamados Jogos Conectados (*Connected Games*). Nestes jogos, a jogabilidade básica não requer uma conexão permanente com outras entidades, como servidores ou outros jogadores. Porém, 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. Embora não haja uma interação direta entre jogadores, o que de certa forma acaba comprometendo o uso do termo "multiusuário" para tal cenário, 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 jogo que segue esta idéia é *Meu Big Brother* [MG05].

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. Porém, estes jogos 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 com 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 a esta audiência. Esta tendência se confirma com a concepção de jogos que unem características de jogos móveis e MMGs, os chamados Jogos Multiusuário Massivos Móveis, do inglês, *Massively Multiplayer Mobile Games (3MG)*.

Estes jogos 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 são atenuados através de projetos adequados às limitações existentes, como o uso de batalhas por turnos ou retardo na execução das ações dos jogadores, bem como uso de recursos específicos das redes de telefonia móvel, como o uso de *Short Message Service (SMS)*. Exemplos de 3MGs são *Samurai Romanesque* [Kri03], *Tibia Micro Edition* [Nok03] e *Pocket Kingdom* [Nok].

3.4 Jogos Pervasivos

Estes jogos aproveitam a visão de Computação Pervasiva para criar novas formas de interação e entretenimento que vão além daquelas realizadas em frente a computadores ou consoles de video-game, permitindo que jogadores desloquem-se e interajam de diferentes formas, através de diversos dispositivos e tecnologias de comunicação.

Este novo paradigma computacional procura promover uma computação melhor integrada ao cotidiano de seus usuários. Sistemas de computação pervasiva propõem o acesso constante do usuário às suas aplicações e dados via diferentes dispositivos e diferentes redes de comunicação sem fio, onde dados contextuais são monitorados pelas aplicações ou por sistemas de suporte, que modificam seu comportamento em função de mudanças relevantes nestas informações, para melhor atender as requisições do usuário. Baseado nestas características, um jogo pervasivo é definido como "um jogo que está sempre presente, disponível aos seus usuários. Os jogadores utilizam diferentes dispositivos e diferentes redes de acesso. O jogo utiliza informações contextuais para aumentar a experiência do usuário em relação à participação no jogo" [BHLM02].

Da mesma forma como concordamos que a computação pervasiva é uma evolução da computação móvel, podemos dizer que jogos pervasivos são inovações a partir dos atuais jogos móveis. 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. Estes jogos são concebidos com base em três tecnologias principais: *(i)* dispositivos móveis, *(ii)* comunicação sem fio e *(iii)* tecnologias capazes de capturar informações no contexto do mundo real dos jogadores, principalmente sua localização física.

Em realidade, é preciso fazer um ressalva ao uso do termo "jogo pervasivo" no contexto desta proposta. Vários jogos que não se baseiam nas tecnologias citadas anteriormente também são classificados como jogos pervasivos. Alguns deles dizem respeito a jogos que utilizam computadores pessoais, mas que usam meios de comunicação do cotidiano como *e-mail*, telefone e fax, para fornecer informações do jogo aos jogadores.

Outros são aqueles que utilizam modos não convencionais para interação homem-máquina entre jogador e jogo. Na proposta de seu precursor, Mark Weiser, o uso de interfaces naturais que facilitem a comunicação do usuário com seu ambiente computacional é apontado como um importante passo em direção à computação ubíqua [MI01]. Estas interfaces devem se aproximar da forma como seres humanos interagem com o mundo real. Esta idéia segue o conceito de Interfaces Tangíveis [IU97], sistemas relativos ao uso de artefatos físicos como representações e controles de sistemas de informação. Um exemplo recente deste conceito é o novo console da Nintendo, o Wii[Nin06]. Este *videogame* captura os movimentos do jogador e as interpreta como ações no jogo.

Neste texto, jogos pervasivos seguem a idéia de utilização de dispositivos móveis e informações do contexto do mundo real do jogador como elementos da dinâmica do jogo. Existem hoje pesquisadores que já realizam estudos na área, através de provas de conceito, protótipos e experiências comerciais [CFG+03] [FBA+03]. Suas possibilidades de aplicação em setores como educação e turismo são inúmeras. Alguns especialistas afirmam que jogos pervasivos têm inclusive o potencial de atrair pessoas que não sejam interessadas em jogos de computador.

Jogos pervasivos oferecem novas oportunidades para projetistas, mas também demandam projetos diferentes dos tradicionais jogos eletrônicos, de maneira análoga a como um sistema computacional pervasivo é diferente de sistemas tradicionais.

Boa parte das experiências com jogos pervasivos realizadas revive jogos infantis clássicos, no estilo "pega-pega" ou "capturar a bandeira". Estes jogos exigem o deslocamento dos jogadores em algum ambiente físico, sendo chamados então de Jogos Baseados em Localização.

Experiências com estes jogos incluem a utilização tanto de esquemas de posicionamento em redes locais sem fio, quanto o uso de serviços de localização oferecidos por operadoras de telefonia celular. No caso destes últimos, os jogos oferecem ainda suporte a um alto número de jogadores interagindo 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 à comunicação em grupo para aprendizado, socialização e atividades culturais.

Muitos experimentos relacionados com jogos pervasivos utilizam também conceitos de realidade virtual, como utilização de dispositivos como capacetes ou utensílios que permitem mesclar objetos do mundo virtual na visão dos jogadores. Algumas experiências com jogos pervasivos são o *Human Pacman* [CFG+03], apresentado na

Figura 4, e *Can You See Me Now?* [FBA+03]. As primeiras experiências comerciais destes jogos datam do final da década de noventa, onde os caso de maior sucesso são relativos aos mercados europeu e asiático, onde a cultura de jogos é mais forte. Exemplos destes jogos são *Botfighters* [Day], *Undercover* [Tea] e *Alien Revolt* [MC05].

Figura 4 – Jogo Human Pacman

3.5. Jogos Massivos Multiusuário Multiplataforma

Embora jogos móveis massivamente multiusuário e jogos baseados em localização ainda estejam em um estágio embrionário, cenários ainda mais inovadores já começam a ser propostos para a união entre jogos multiusuário e dispositivos móveis. Um deles seria a idéia de jogos multiplataforma que permitissem o acesso ao mundo virtual através de diferentes dispositivos. Enquanto os atuais 3MGs restringem seu uso a dispositivos móveis, jogos multiplataforma permitiriam o acesso do jogador tanto via computadores pessoais, quanto via telefones celulares.

Porém, é preciso ressaltar que as diferentes condições de acesso de dispositivos móveis e computadores obrigam que o projeto destes jogos deva ser muito bem concebido, permitindo uma interação justa entre jogadores através de diferentes dispositivos. Uma idéia simples é permitir diferentes funcionalidades de acordo com o dispositivo de acesso do jogador. Por exemplo, um jogador poderia utilizar seu telefone celular para oferecer um artefato em algum mecanismo de troca disponibilizado pelo jogo, ou enviar mensagens para outros jogadores que estivessem conectados via computadores pessoais. Como frisado na introdução deste documento, o suporte à mobilidade é apontado como requisito imprescindível para o sucesso de jogos futuros [Wal04]. Neste sentindo, esta visão começa a ser incorporada em alguns jogos, como *Lineage Mobile* [NCs05b], *Mogi, Item Hunt* [NG] e *HinterWars: The Aterian Invasion*.

4. Fatores limitantes para jogos multiusuário

Apesar de diferentes cenários de aplicações, todos possuem uma preocupação comum. 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 confiável e, possivelmente, de forma rápida e confiável. Além disso, tais ações precisam também ser ordenadas de modo a garantir a correta execução dos eventos no jogo, caso contrário, situações indesejadas podem acontecer como um jogador interagindo com outro já ausente do jogo ou coletando um objeto previamente retirado por outro jogador. Porém, a separação física dos componentes e jogadores decorrente da natureza distribuída de ambientes

virtuais em rede impõem obstáculos para a manutenção de um estado global consistente entre os jogadores.

Além de trazer problemas inerentes de aplicações de espaço compartilhando, cada um dos diferentes cenários apresentados na seção anterior incorpora ainda novos problemas específicos às suas características. Por exemplo, no caso de MMGs, a necessidade de escalabilidade face ao alto número de usuários cria mais um problema complexo a ser resolvido.

Helin [Hel03] aponta três grandes problemas enfrentados por jogos multiusuário em rede. Primeiramente, a infra-estrutura 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 nas subseções seguintes.

4.1. Plataforma Física

Smed [SH02] nomeia a infra-estrutura de rede utilizada como a *plataforma física* de suporte ao jogo. 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 das redes geográficas, como a 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 a serem vistas a seguir (Seção 4.2).

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. Por exemplo, atrasos para a propagação e retransmissão dos sinais elétricos necessários para uma transmissão de uma mensagem entre Europa e Estados Unidos variam entre 25 a 30 ms. Adicionando-se o tempo decorrente das rotinas de roteamento, enfileiramento e manipulação de pacotes, o atraso médio desta transmissão eleva-se à faixa de 70 a 90 ms [SKH01].

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.

4.2. Plataforma Lógica

Além das limitações impostas pela infra-estrutura 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 (ou ausência dela) de um jogo. Enquanto não há muito a ser feito em relação à plataforma física (a não ser investir em *hardware* e rede), a plataforma lógica assume um papel fundamental para atenuar os efeitos prejudiciais que a plataforma física pode impor, como será visto na Seção 5.2.

4.3. 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 *on-line* 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 *on-line*. Este relaciona-se com a possibilidade de jogadores trapaceiros utilizarem-se de recursos ou meios anti-éticos no intuito de obter vantagens ilegais sobre os demais jogadores. Para jogos multiusuário, jogadores trapaceiros criam desequilíbrio, desencorajando os demais jogadores, principalmente os iniciantes, a participarem do jogo. Em um mercado onde o jogador deve ser estimulado a permanecer o maior tempo possível no jogo, a facilidade de trapacear pode arruinar o sucesso 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 onze 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 por administradores de jogos *on-line*. 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 on-line, 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 *on-line*. Nesta abordagem, ressaltam-se tanto medidas simples como o uso de políticas para definições de senhas dos jogadores, quanto mais complexas e controversas, como o uso de análises estatísticas para detecção de jogadores que sejam bons demais para não estar trapaceando.

GauthierDickey *et al* [GZL05] categorizam trapaças de acordo com o nível onde as mesmas ocorrem: na rede, nos protocolos, nas regras do jogo ou na própria aplicação. No nível do jogo, trapaças ocorrem por ambiguidades ou por falhas ou manipulação nas regras do jogo para se obter resultados não planejados. A prevenção deste tipo de trapaça requer uma revisão do projeto do jogo de modo a resolver tais falhas. Trapaças no nível de rede ocorrem por problemas de segurança de redes, como ataques por Negação de Serviço (DoS). Trapaças nos níveis da aplicação e de protocolos ocorrem por alterações no código do *software* cliente do jogo, fazendo com que o jogador trapaceiro obtenha vantagens como: um tempo maior de reação ao jogador trapaceiro, ou acesso a informações que o mesmo não deveria ter.

5. Soluções Clássicas para Suporte a Jogos Multiusuário

De acordo com Smed [SKH01], os conceitos apresentados na Seção 4.1 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 permitem a definição de arquiteturas de comunicação, de controle e dados, que juntamente com técnicas compensatórias podem ajudar a atenuar os efeitos destes limitadores.

5.1 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 5 apresenta as principais formas de distribuição das mensagens entre os jogadores de um jogo multiusuário.

A primeira técnica, conhecida como *broadcast* (Figura 5 (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.

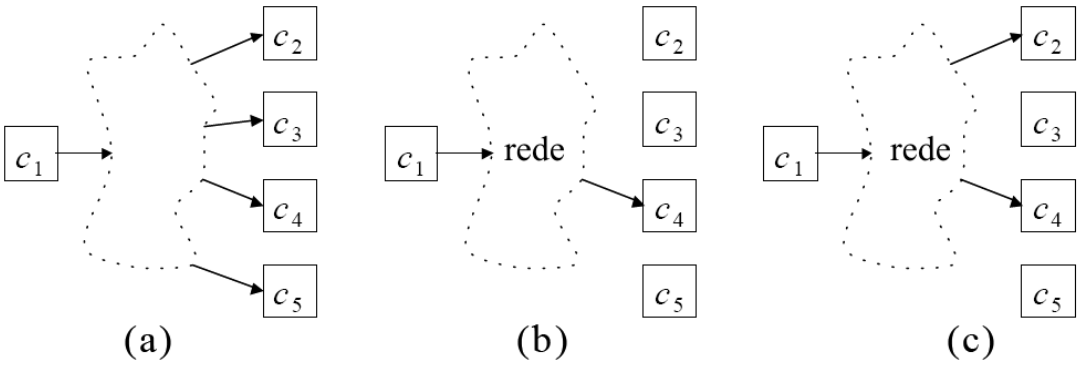


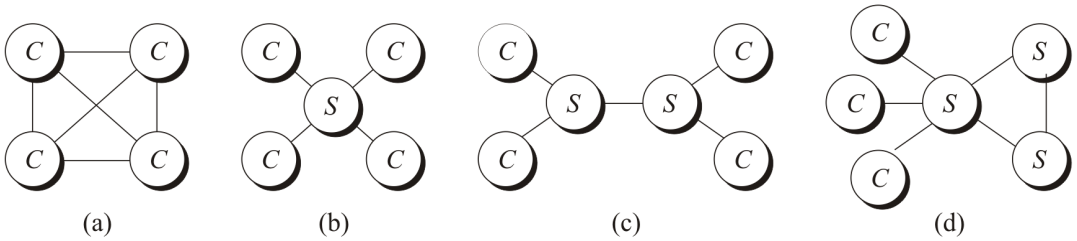
Figura 5 – Técnicas para transmissão de mensagens. (a) Broadcast, (b) Unicast e (c) Multicast

Uma segunda técnica, chamada *unicast*, representada pela Figura 5(b), determina que cada mensagem possua um único emissor e um único receptor. Embora sendo mais eficiente que *broadcast*, torna-se desinteressante quando da necessidade de envio de uma única mensagem para múltiplos destinatários, 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 5(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 infra-estrutura da Internet para sua utilização, bem como questões de segurança relacionadas com implementação de *multicast* sobre uma rede TCP/IP.

5.2. 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 6 ilustra as mais conhecidas arquiteturas para jogos multiusuário [Hel03].



A Figura 6(a) apresenta a arquitetura ponto a ponto, do inglês *Peer-to-Peer (P2P)*. 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. Desta forma, cada nó precisa se comunicar com todos os 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.

Na arquitetura cliente/servidor, representada pela Figura 6(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 *on-line* 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 6(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 6(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.

5.3 Técnicas compensatórias

As arquiteturas apresentadas nas subseções anteriores por si só não são suficientes para reduzir os requisitos de comunicação e recursos necessários ao suporte de um jogo multiusuário. Porém, algumas técnicas já existentes para ambientes virtuais em rede podem ser aplicadas como forma de atenuar a escassez de algum destes recursos. A seguir serão apresentadas algumas dessas técnicas.

5.3.1. 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 utilizado ao custo de processamentos e atrasos adicionais pela execução de tais procedimentos.

5.3.2 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 ou extrínseco. 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.

5.3.2 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 de técnicas de previsão para diminuir o impacto da ausência de informações de atualização. Para isto, Dead Reckoning, técnicas navegacionais que utilizam estimativas de cálculo para 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. No 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.

7. Uma arquitetura simplificada para jogos multiusuário

Para melhor ilustrar o funcionamento de um jogo multiusuário, esta seção apresenta uma arquitetura simples para desenvolvimento de jogos no modelo cliente/servidor para possibilitar o compartilhamento do estado do jogo entre jogadores. Esta arquitetura subdivide-se em duas partes: um *framework* para comunicação entre jogadores e o servidor do jogo e outro *framework* para a simulação (processamento do estado) do jogo.

7.2 *Framework* de comunicação

Nesta arquitetura, procura-se uma forma de padronizar a comunicação entre jogadores e o servidor do jogo. Assim sendo, o fluxo de comunicação segue um modelo simplificado para aplicações de espaço compartilhado, apresentado na Figura 6.

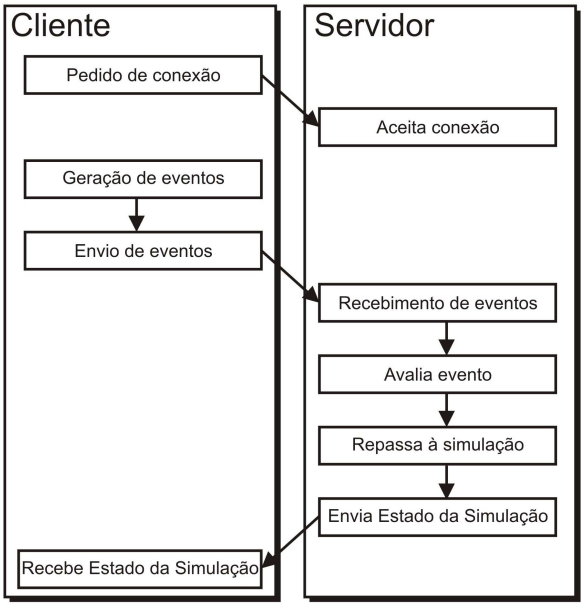


Figura 6 – fluxo de comunicação entre cliente/servidor

A partir da conexão do cliente com o servidor, o cliente começa a gerar eventos de acordo com a interação do usuário com a interface do jogo. Estes eventos são então repassados ao servidor do jogo. O servidor lê, realiza o parsing das mensagens recebidas e repassa cada evento ao *framework* de simulação. A simulação também é responsável pela geração de novos eventos, de acordo com a lógica do jogo, de forma independente das ações dos clientes. A partir da simulação, o estado do jogo é então repassado a cada jogador, de acordo com sua área de interesse. O cliente recebe esta informação, que então a apresenta ao usuário. A partir desta representação, o ciclo se repete.

Uma vantagem nesta abordagem é que se procura evitar *(i)* processamentos locais custosos nos dispositivos, *(ii)* a necessidade de uma constante comunicação para repasse dos eventos, bem como *(iii)* a necessidade de procedimentos mais complexos de sincronização entre clientes e o servidor. Em suma, os clientes atuam como interfaces de apresentação e interação com o servidor, sem processamento do estado do jogo localmente. Todo processamento relativo à simulação do mundo virtual é de responsabilidade do servidor. Em contrapartida, a quantidade de informações que representa o estado do jogo tende a ser maior que o repasse do conjunto de eventos enviados pelos jogadores. Nesta proposta este problema é atenuado, pois o mundo virtual será dividido em áreas de interesse, e o jogador só recebe informações sobre a área com que interage em um dado instante.

Baseado nesta idéia, o *framework* de comunicação utiliza a proposta de Bracken el al[BBV03]. Nesta proposta, a comunicação utiliza a abordagem de retransmissão dos eventos pelo servidor apenas para clientes que utilizem computadores pessoais.

Outro aspecto importante é que toda a comunicação entre clientes e o jogo é encapsulada na forma de eventos de jogo (GameEvent). Toda mensagem transmitida pelos clientes é transformada pelo *framework* de comunicação para um formato padronizado, antes de ser repassado ao *framework* de simulação. Este formato é representado pela Figura 7.

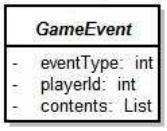


Figura 7 – Diagrama UML para um Evento do Jogo

Cada evento é gerado por um jogador (playerId), e possui um tipo que o identifica dentro do protocolo de comunicação com o servidor, e também com a simulação (eventType). Embora o formato de dados relacionado com cada evento seja dependente do projeto de cada jogo, o mesmo deve ser transparente para o *framework*, uma vez que é disponibilizada esta interface para eventos e seu tratamento pelo servidor. Cada jogo pode ter seu formato distinto. Para isto, cada evento utiliza uma lista de conteúdo que guarda o conteúdo de cada mensagem transmitida entre jogadores e o servidor, e vice-versa. Por exemplo, o evento relativo à movimentação do avatar do jogador encapsula as novas coordenadas do mundo virtual para onde o mesmo queira se deslocar.

A Figura 8 apresenta uma versão simplificada da arquitetura do *framework* de comunicação. A classe GameServer representa o servidor do jogo. Este tem a função de aceitar as conexões dos clientes, gerenciar conexões ativas de jogadores em computadores pessoais, gerenciar a classe responsável pelo jogo em si, chamada GameController. O servidor lida com todo o processo *marshaling* e *unmarshaling* de mensagens transmitidas entre jogadores e o servidor do jogo, criando uma transparência no processo de comunicação.

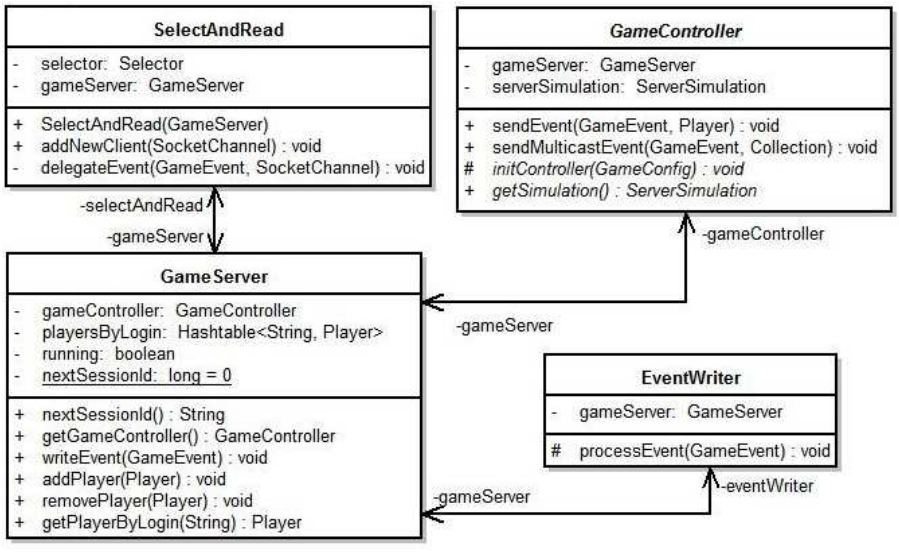


Figura 8 – Arquitetura do framework de comunicação

A estrutura de comunicação do Servidor possui duas classes auxiliares: SelectAndRead e EventWriter. A primeira classe é responsável pelo processamento da mensagem recebida pelo servidor, sua transformação em um evento do jogo (GameEvent), e repasse ao GameController.

A segunda é responsável por receber as informações do estado do jogo, encapsulado em um evento (GameEvent) e repassá-los aos jogadores. A classe abstrata GameController representa o processo servidor do jogo. Um jogo deve estender esta classe, inserindo a lógica relativa a sua simulação. Esta classe possui uma referência à simulação do jogo, repassando-lhe todos os eventos recebidos pelo servidor.

7.2. *Framework* de Simulação

O segundo componente da arquitetura é o *framework* de simulação. Este componente é o responsável por manter consistente o estado da simulação do jogo. Para isto, este componente recebe os eventos referentes aos comandos de cada jogador, que já foram processados pelo *framework* de comunicação. Estes eventos são postos em uma fila, onde o estado corrente do mundo é modificado de acordo com sua execução.

O estado do jogo engloba o estado de cada elemento que forma o mundo virtual, quer sejam estes objetos, NPCs ou personagens controlados por jogadores. Para modelar este componente, adaptou-se a proposta de Alexander Thor [Ale02] [Ale03a] [Ale03b] de um *framework* de simulação para MMGs. Aqui, todo objeto do jogo é representado por uma classe GameObject, apresentado na Figura 9.

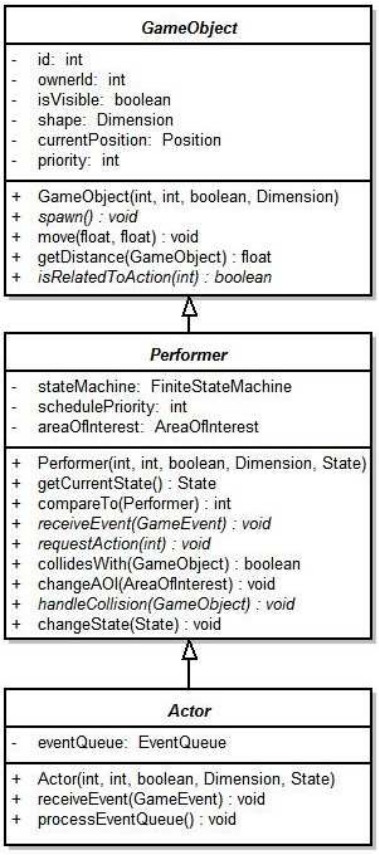


Figura 9 – Diagrama de Classes – Objetos do Jogo

Estes objetos incluem desde itens até personagens do jogo, sendo que cada objeto possui um identificador único na simulação – atributo id. O *framework* diferencia certos tipos de objetos através de especializações. A classe abstrata Performer representa elementos do jogo que realizam ações (atuadores), como *Non-player Characters* (NPC). Cada atuador tem seu comportamento controlado por uma máquina de estados finitos, que indica qual o estado atual do objeto, bem como as transições entre os possíveis estados que o elemento pode assumir.

Um atuador possui associada uma área de interesse, que representa o conjunto de objetos que o mesmo pode interagir em um dado instante. A classe abstrata Actor representa um personagem controlado por um jogador, aqui chamado de ator. Um ator também é capaz de interagir com a simulação. Cada ator possui uma lista de eventos que acumula os comandos enviados pelo usuário, de acordo com sua interação com a aplicação-cliente do jogador.

Para gerência da criação e da referência aos objetos do jogo, são utilizadas as classes GameObjectFactory e GameObjectManager, apresentados na Figura 10. A primeira estabelece uma forma padronizada para criação dos objetos, que utilizando o padrão de projeto Factory.

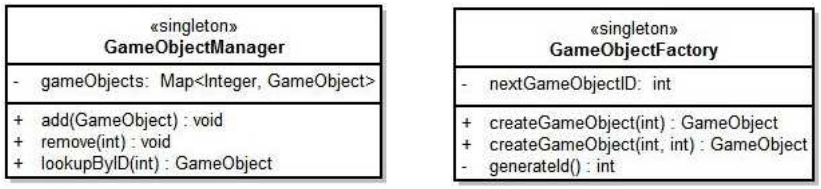


Figura 10 – Diagrama de Classes – Gerenciadores de Objetos do Jogo

Este objeto é o responsável por atribuir um código único na simulação a cada objeto criado no jogo. A segunda classe encapsula a função de um dicionário de todos os objetos do jogo. Sua principal função é resolver referências a objetos do jogo, de acordo com o identificador do objeto.

Como adaptação do *framework* original [Ale02] [Ale03a] [Ale03b], foi criada uma classe chamada AreaOfInterest – (Figura 11), que representa o conceito de área de interesse. Esta classe possui um conjunto de objetos, NPCs e atores de uma área do mundo virtual. Esta área pode representar diferentes representações geométricas na topologia de um mundo virtual, como áreas retangulares ou circulares. Em sua principal função, é possível determinar se a partir de uma posição absoluta do mundo virtual, esta encontra-se dentro de uma determinada área de interesse. Nesta proposta, o mundo virtual será dividido em várias regiões. De modo a gerenciar estas áreas, criou-se um gerenciador de áreas de interesse, representado pela classe abstrata AOIManager. Sua principal função é descobrir qual a área de interesse, a partir de uma posição absoluta do mundo virtual.

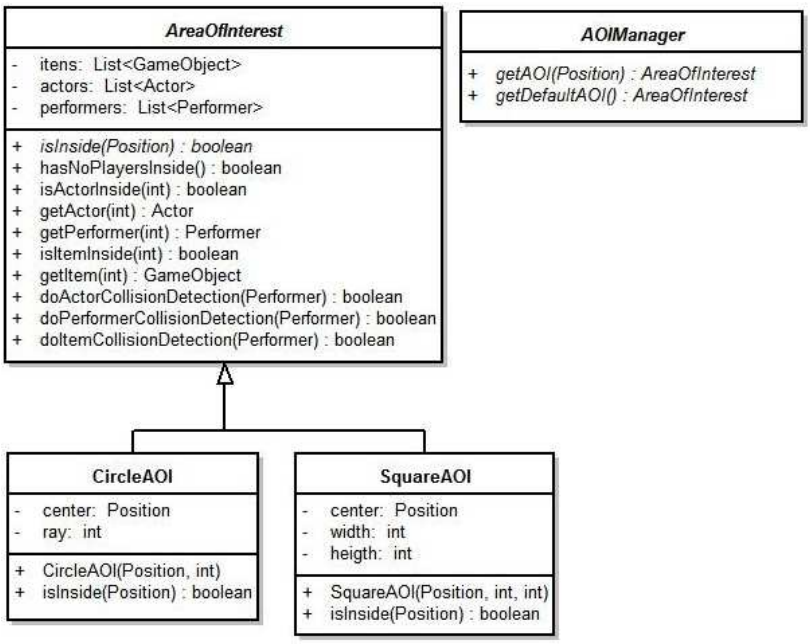


Figura 11 – Diagrama de Classes – Area de Interesse

O escalonamento de ações do jogo é realizado pela classe SchedulerManager, apresentado na Figura 12. Este objeto executa a ação do estado corrente de cada atuador e ator ativo na simulação. Este possui métodos para inserir (scheduleTask) ou remover (cancelTask) objetos na simulação do jogo. O método processTasks recebe como argumento um intervalo de tempo durante o qual, executa tarefas pendentes em todos os objetos que estejam ativos na simulação, de acordo com suas prioridades.

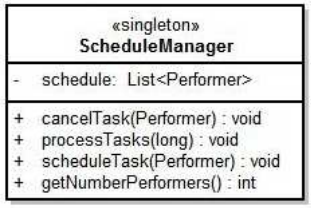


Figura 12 - Diagrama de Classes - Escalonador

Para gerenciar todos estes elementos, o *framework* oferece uma classe chamada ServerSimulation (Figura 13) que possui referências a todos elementos gerenciadores da simulação, no caso, os gerenciadores de objetos, o escalonador e o gerenciador das áreas de interesse.

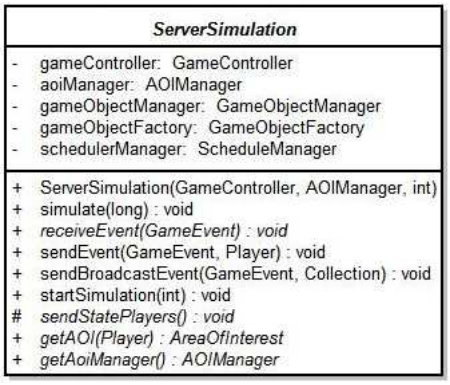


Figura 13 – Diagrama de Classes – Servidor da Simulação

Esta classe possui o método simulate, responsável por executar ações em todos os atuadores ativos no jogo em um dado instante. Este método basicamente delega a execução da simulação à classe SchedulerManager.

7. Conclusões

Jogos Multiusuários é um tema cada vez mais explorado tanto na academia, quanto na indústria. Suas aplicações hoje não são apenas restritas ao entretenimento, mas também à educação e treinamento em diversas áreas do conhecimento. Esperamos que através deste tutorial, o leitor possa ter se familiarizado com os principais aspectos relacionados com o tema, seus requisitos e técnicas de desenvolvimento destas aplicações.

8. Referências

[BE05] Blizzard Entertainment. Starcraft, 2005. Disponível em http://www.blizzard.com/starcraft/. Acesso em 02 de Maio de 2007.

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

[Ale03a] Thor Alexander. Massively Multiplayer Game Development (Game Development), volume 1, chapter 2.1: *Building a Massively Multiplayer Game Simulation Framework, Part 1: Structural Modeling*, pages 103 – 114. Charles River Media, Inc., Rockland, MA, USA, 2003.

[Ale03b] Thor Alexander. Massively Multiplayer Game Development (Game Development), volume 1, chapter 2.1: *Building a Massively Multiplayer Game Simulation Framework, Part 1: Behavioral Modeling*, pages 115 – 122. Charles River Media, Inc., Rockland, MA, USA, 2003.

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

[BE06] Blizzard Entertainment. The World of Warcraft Community Site,

2006. Blizzard, Inc. Disponível em http://www.worldofwarcraft.com/. Acesso em 02 de Maio de 2007.

[BHLM02] S. Björk, J. Holopainen, P Ljungstrand, and R Mandryk. Special issue on ubiquitous games. *Personal and Ubiquitous Computing*, 6(5 – 6):358 – 361, December 2002.

[Blu04] Bluetooth.org. The Official Bluetooth Membership Site, 2004. Disponível em http://www.bluetooth.org/spec/. Acesso em 02 de Maio de 2007.

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

[Cen01] The Norwegian Computing Center. White paper – Massive Multiplayer Game Networking. Technical report, The Norwegian Computing Center, 2001.

[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.

[CG04] V. Lo C. GauthierDickey, D. Zappala. A fully distributed architecture for massively multiplayer online games. In *SIGCOMM 2004 Workshops: Proceedings of ACM SIGCOMM 2004 workshops on NetGames '04*. ACM Press, 2004.

[Day] Daydream. Bot Fighters. Disponível em http://www.botfighters.com. Acesso em 02 de Maio de 2007.

[Eve06] Everquest. Everquest Live.com, 2006. Disponível em http://everquest.com. Acesso em 02 de Maio de 2007.

[FBA+03] Martin Flintham, Steve Benford, Rob Anastasi, Terry Hemmings, Andy Crabtree, Chris Greenhalgh, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Where on-line meets on the streets: experiences with mobile mixed reality games. In *CHI '03: Proceedings of the conference on Human factors in computing systems*, pages 569–576. ACM Press, 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.

[GZL05] Chris GauthierDickey, Daniel Zappala, and Virginia Lo. Peer-to-peer support for Massively Multiplayer Games. In *Proceedings of the 24th*

Conference of the IEEE Communications Society (In Submission). IEEE Computer Society, March 2005.

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

[IDG02] IDGA. IGDA online games white paper, 2002. IDGA, http://www.igda.org/online/IGDAOnlineGamesWhite_paper_2002.pdf . Acesso em 02 de Maio de 2007.

[IEE05] IEEE-802.11. IEEE 802.11, The Working Group Setting the Standards for Wireless LANs, maio 2005. http://grouper.ieee.org/groups/802/11/.

[IU97] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM Press.

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

[MC05] M1nd Coorporation. Alien Revolt, Maio 2005. Disponível em http://www.alienrevolt.com/pt. Acesso em 02 de Maio de 2007.

[MGS04] Microsoft Game Studios. Welcome to mythica, 2004. http://mythica.com. Acesso em 02 de Abril de 2004.

[MI01] R.L. Mandryk and K.M. Inkpen. Supporting free play in ubiquitous computer Games. In *Proceedings UbiComp 2001 Workshop on Ubiquitous Gaming*, October 2001.

[Mic04] Microsoft. Microsoft Game studios | the official age of empires website, 2004. Disponível em http://www.microsoft.com/games/empires/. Acesso em 02 de Maio de 2007.

[NCs05a] NCsoft. Lineage 2, The Chaotic Chronicle – the official web site, 2005. NCsoft, Inc. http://www.lineage2.com. Acesso em 02 de Maio de 2007.

[NCs05b] NCsoft. Lineage, Dark Conquest – the official web site, 2005. NCsoft, Inc. http://www.lineage.com. Acesso em 02 de Maio de 2007.

[NG] Newt Games. Mogi item hunt. Disponível em http://www.mogimogi.com. Acesso em 02 de Maio de 2007.

[Nin05] Nintendo. Nintendo's official site for nintendo DS, 2005. Disponível em http://www.nintendods.com/. Acesso em 02 de Maio de 2007.

[Nin06] Nintendo. Nintendo's official site for nintendo Wii. Disponível em http://www.nintendowii.com/. Acesso em 02 de Maio de 2007.

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

Acesso em 02 de Maio de 2007.

[Nok03] Nokia. Tibiame Case Study, June 2003. Disponível em http://ncsp.forum.nokia.com/downloads/nokia/documents/Tibia_v_1_0.pdf. Acesso em 02 de Maio de 2007.

[Nok04] Nokia. Nokia N-Gage | home, 2004. Nokia N-Gage. http://www.n-gage.com. Acesso em 02 de Maio de 2007.

[Ori02] Origin. ORIGIN Ultima On-line, 2002. Disponível em http://www.uo.com.

[Pri00] M. Pritchard. How to hurt the hackers: The scoop on internet cheating and how you can combat it, July 2000. Gamasutra, http://www.gamasutra.com/features/20000724/pritchard01.htm. Acesso em 02 de Maio de 2007.

[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.

[Rag05] Ragnarok. RAGNAROK online, 2005. Disponível em http://iro.ragnarokonline.com/. Acesso em 02 de Maio de 2007.

[SE06] Sony Entertainment. Playstation portable – PSP, 2006. Disponível em http://www.us.playstation.com/psp.aspx. Acesso em 02 de Maio de 2007.

[SH02] Kaukoranta Smed and Hakonen. A review on networking and multiplayer computer games. Technical Report 454, Turku Centre for Computer Science, 2002.

[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.

[SM05] Sun Microsystems. Java 2 Platform, Micro Edition, 2005. Disponível em http://java.sun.com/j2me/. Acesso em 02 de Maio de 2007.

[Ste03] Steam. Counter-strike.net – the official web site, 2003. Counter-Strike.net, Inc. Disponível em http://www.counter-strike.net/. Acesso em 02 de Maio de 2007.

[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.

[Tea] YDream Team. Undercover. Disponível em http://hk.playundercover.com. Acesso em 02 de Maio de 2007.

[Wal04] GordonWalton. Nine things to look for in the next generation of MMOG, 2004. Disponível em http://www.gdconf.com/archives/2004/walton_gordon.ppt. Acesso em 02 de Maio de 2007.

[Woo06] Bruce Sterling Woodcock. An analysis of MMOG subscription growth, 12.0, June 2006. Disponível em http://www.mmogchart.com/. Acesso em 02 de Maio de 2007.

[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.