

wGEM: um *Framework* de Desenvolvimento de Jogos para Dispositivos Móveis

Carlos A. C. Pessoa¹, Geber Lisboa Ramalho¹, André Luiz Battaiola²

¹ Centro de Informática - Universidade Federal de Pernambuco (UFPE), Caixa Postal 7851 - 50732-970 - Recife - PE

² Departamento de Computação - Universidade Federal de São Carlos (UFSCar), Via Washington Luiz, Km. 235, Caixa Postal 676 - 13565-905 - São Carlos - SP.

{cacp;glr}@cin.ufpe.br, andre@dc.ufscar.br

Resumo. *A recente introdução de plataformas de software para dispositivos móveis, como o Java 2 Micro Edition (J2ME), criou um cenário inteiramente novo para o desenvolvimento de aplicações embarcadas. Entre as aplicações-chave nesse mercado estão os jogos eletrônicos, dado o enorme sucesso do mercado de jogos para PCs. Entretanto, dada a simplicidade de J2ME e seu curto tempo de criação, várias ferramentas de apoio à implementação de jogos precisam ser concebidas, em especial um framework de desenvolvimento de jogos. Este trabalho apresenta o framework wGEM, resultado de uma generalização e adaptação de frameworks de desenvolvimento de jogos para PCs às limitações dos dispositivos móveis e da tecnologia J2ME.*

Abstract. *The recent introduction of software development platforms for mobile devices, such as Java 2 Micro Edition (J2ME), has created a whole new scenario for third party development of embedded applications. Among the most prospective applications for this new market, wireless games have received a special attention given the tremendous success of the PC games industry. However, given the J2ME's simple resources and its recent emergence, support tools to the development of games need to be designed, in special a framework for game development. This paper introduces wGEM, a pioneer game engine that adapts the features found on PC-based game engines to the limitations of the mobile devices and the J2ME technology.*

1. Introdução

Um dos maiores exemplos de convergência digital é a adoção pelos diversos dispositivos móveis atuais como PDAs, celulares e *paggers* do modelo PC de desenvolvimento, instalação e uso de aplicativos. Há cerca de 5 anos esses aparelhos eram totalmente fechados, pois a comunidade em geral não era capaz de desenvolver e nem instalar novos programas em tais dispositivos. No entanto, há uns 3 anos os maiores fabricantes de sistemas embarcados e empresas de software vem buscando meios de abolir esse modelo fechado, observando o sucesso da indústria de software para PCs.

Um dos mais recentes e maiores esforços para a implementação dessa idéia é a especificação e a implementação da tecnologia Java 2 Micro Edition (J2ME) [3], resultado da adaptação da tecnologia Java para as limitações dos sistemas embarcados.

Entre as aplicações-chave nesse novo mercado de software estão os jogos eletrônicos, dado o enorme sucesso do mercado de jogos para PCs. Entretanto, dada a simplicidade de J2ME e seu curto tempo de criação, várias ferramentas de apoio à implementação de jogos precisam ser concebidas, em especial um *framework* de desenvolvimento de jogos, ferramenta fundamental para uma produção em escala industrial.

Neste artigo apresentamos o *framework* wGEM (de Wireless Game Engine for Mobile Devices), um esforço pioneiro de adaptação das tecnologias de desenvolvimento de jogos para PCs às limitações encontradas em sistemas embarcados e na tecnologia J2ME. O wGEM já vem sendo utilizado com sucesso no CESAR [13] para a produção em escala profissional de jogos eletrônicos baseados em J2ME.

Na seção 2, nós ressaltamos o enorme potencial da área de jogos eletrônicos no contexto de convergência digital. Na seção 3, apresentamos os principais conceitos da tecnologia J2ME, mostrando que, apesar de algumas limitações, essa tecnologia pode ser utilizada para o desenvolvimento de jogos. A seção 4 detalha o conceito de *framework* de jogos a partir dos seus requisitos. Na seção 5, é apresentado em maiores detalhes o *framework* wGEM, os seus requisitos e sua arquitetura. Na seção 6, apresentamos alguns estudos de caso de uso do wGEM e, finalmente, na seção 7, comentamos nossas conclusões e direções de trabalhos futuros.

2. A Era dos Jogos para Dispositivos Móveis

Pesquisas estimam que mais de 80% das pessoas da Europa utilizarão celular em 2005, e que mais de 1 bilhão de pessoas em todo o mundo utilizarão Internet em dispositivos móveis em 2003 [8]. Tais previsões, aliadas com o enorme sucesso comercial dos jogos eletrônicos para PCs, demonstram o potencial mercadológico dos jogos para dispositivos móveis.

Hoje, a diversidade de equipamentos eletrônicos portáteis em que é possível a utilização de jogos é imensa. Diversos modelos de PDAs, celulares, pagers, videogames portáteis e vários outros tipos de equipamentos são lançados a cada dia, permitindo esta forma de entretenimento. Note-se que, apesar dessa diversidade, a comunidade de desenvolvedores não pode, na maioria dos casos, construir aplicativos para tais aparelhos. Apenas os próprios fabricantes dos dispositivos podem fazê-lo.

No entanto, após vários anos de arquiteturas fechadas e incompatíveis, os fabricantes de dispositivos como celulares, pagers e PDAs uniram-se para viabilizar o desenvolvimento de aplicações para esses dispositivos da mesma forma como ele ocorre nos computadores pessoais. Em PCs, um software é desenvolvido para um dado sistema operacional, ou máquina virtual como ocorre com Java, e é então capaz de ser executado em qualquer computador, não importando o fabricante do mesmo, bastando a presença do sistema operacional e/ou máquina virtual necessária.

O resultado desse esforço, liderado pela Sun Microsystems e empresas como Motorola, Nokia, Ericsson e Samsung, foi o surgimento e padronização da tecnologia

Java 2 Micro Edition como meio de desenvolvimento em larga escala de aplicativos para dispositivos móveis.

Outras tecnologias com esse mesmo objetivo também surgiram, como o BREW (Binary Runtime Environment for Wireless) [15] da Qualcomm. Apesar de apresentar algumas vantagens, como maior eficiência e melhor uso de memória, BREW tem ainda futuro incerto. De fato, BREW é uma camada de software (API) voltada exclusivamente para os aparelhos contendo chips da QUALCOMM, todos baseados em na tecnologia CDMA, que representa 20% do mercado. J2ME, por outro lado, já está tendo uma grande aceitação, com sua máquina virtual sendo embutida em dezenas de aparelhos dos diversos fabricantes [16].

3. Java 2 Micro Edition (J2ME)

Atualmente existem três edições diferentes da linguagem Java – Enterprise Edition (J2EE) [1], Standard Edition (J2SE) [2] e Micro Edition (J2ME) [3], cada uma sendo direcionada para diferentes categorias de dispositivos. A linguagem Java 2 Micro Edition (J2ME), com aproximadamente dois anos de criação, é uma simplificação da linguagem Java 2 Standard Edition (J2SE) realizada através da remoção e modificação de partes fundamentais de J2SE com o objetivo de criar um ambiente de execução de programas para dispositivos com grandes restrições de memória e processamento. [9]

3.1. Arquitetura de J2ME

Uma grande parte do trabalho de definição da arquitetura da nova linguagem foi dedicada a considerar que os componentes da linguagem Java original deveriam ser cortados, mantidos ou modificados. Com a observação da variedade de dispositivos envolvidos, J2ME adotou uma arquitetura modular e escalável, consistindo de um conjunto de blocos que se complementam para representar as particularidades dos grupos de dispositivos existentes. J2ME define três camadas de software construídas sobre o sistema operacional presente em cada dispositivo, a saber:

- Camada da Máquina Virtual Java (JVM), que é uma implementação da máquina virtual Java baseada no sistema operacional presente em cada dispositivo;
- Camada de definição de uma Configuração, acima da anterior, define uma plataforma mínima para uma larga categoria de dispositivos;
- Camada de definição de um Perfil (profile), mais próxima dos usuários e desenvolvedores de aplicações, que define a API voltada para demandas específicas de um segmento de mercado, tais como, carros, celulares, televisores, etc.

Baseado nesse conceito de camadas, apenas as funcionalidades comuns a todos os dispositivos fariam parte das camadas mais centrais, enquanto que as particularidades ficariam restritas apenas aos perfis e configurações mais específicas. Naturalmente, um programa escrito para um dado perfil tem a portabilidade garantida para qualquer dispositivo que suporte tal perfil. A idéia é que isso ocorra da mesma maneira que aplicações para PCs são desenvolvidas para um determinado sistema operacional sem preocupação com o fabricante do computador que irá executá-lo. Atualmente existem duas configurações, uma para dispositivos fixos com grande demanda de energia e conexão, Connected Device Configuration - CDC [4], e outra para dispositivos móveis e mais simples, Connected Limited Device Configuration - CLDC [5]. Um perfil que

abrange celulares e pagers, Mobile Information Device Profile – MIDP [6] já está definido sobre o CLDC. Da mesma forma, já existe um perfil sobre o CDC, o Foundation Profile [7]. Outros perfis estão em desenvolvimento, como é o caso do Personal Profile (extensão do Foundation Profile).

3.2. Suporte de J2ME ao desenvolvimento de jogos

Em se tratando do uso de J2ME para o desenvolvimento de jogos, mais precisamente da configuração CLDC e do perfil MIDP, uma observação rápida mostra a viabilidade pela existência das seguintes características:

- Suporte à programação orientada a objetos (OO);
- Suporte à manipulação da tela gráfica para o desenho de imagens e formas geométricas;
- Suporte à manipulação do teclado dos dispositivos, sem o qual seria muito difícil desenvolver um jogo, que é uma aplicação inerentemente interativa;
- Suporte à conectividade com outros dispositivos, possibilitando que jogos em rede sejam desenvolvidos através do protocolo HTTP fornecido pelo perfil MIDP.

Porém, existem certas limitações bastante significativas, como as seguintes:

- Ausência de ponto flutuante, o que obriga o programador a adaptar algoritmos de modelagem física e processamento gráfico;
- Ausência de acesso à cor de um pixel da tela, tornando inviável a execução de algoritmos que dependem do conteúdo gráfico da tela;
- Ausência de som, inviabilizando certos tipos de jogos e tornando outros menos interessantes;
- Baixo poder de processamento, em torno de 25 MHz, inviabilizando a execução de certos jogos;
- Pouca memória, em média 32 KB para execução de programas e 128 KB para persistência de dados, prejudicando jogos que precisam armazenar muitos dados;
- Ausência de métodos de construção e renderização de polígonos, oferecendo este suporte apenas para primitivas gráficas básicas (elipses, retângulos e linhas);
- Ausência de serialização de objetos para, por exemplo, a atualização dos jogos através de carga de objetos sob demanda, ou ainda a troca de objetos para jogos em rede.

O problema do baixo poder de processamento dos dispositivos agrava-se ainda mais em razão dos programas em J2ME serem executados por uma máquina virtual JAVA, o que por si só já torna a execução mais lenta do que se o programa fosse compilado diretamente para a linguagem de máquina. Essa compilação para código nativo ainda não é possível, mas pode tornar-se viável caso os fabricantes dos dispositivos forneçam as ferramentas necessárias.

Diante dessas limitações, o desenvolvimento de jogos para dispositivos móveis impõe uma série de restrições e limitações. No entanto, a utilização de um *framework* pode suprir parte desses problemas, tais como a implementação de mecanismos de desenho de polígonos, utilização de números reais, além dos conceitos específicos de jogos como *sprites*, *tiles*, mapas, etc.

4. Frameworks de Jogos

Como resultado da crescente utilização da engenharia de software no desenvolvimento de jogos para PCs, a indústria começou a isolar os recursos utilizados na maioria dos jogos em *frameworks de desenvolvimento* (também chamados de “motores de jogos” - *game engines*). A idéia principal é permitir que os recursos comuns a quase todos os jogos sejam reutilizados para cada novo jogo criado. Neste caso, a cada novo jogo, se implementa apenas seus requisitos particulares.

A seguir são resumidos os principais requisitos de um *framework* e as considerações necessárias para a implementação de uma ferramenta como esta. Este resumo serve de base para a concepção do wGEM, por este ser um *framework* de desenvolvimento de jogos, embora voltado para dispositivos muito limitados.

4.1. Requisitos

Um *framework* é a peça chave no desenvolvimento de jogos de computador, tendo como principal objetivo executar uma grande parte das tarefas de baixo nível necessárias na maioria dos jogos. Alguns dos principais requisitos de um *framework* são mostrados a seguir [8]:

- Arquitetura modular, para que possa ser utilizado em cada novo jogo;
- Bom nível de abstração, ao prover objetos com funcionalidades já implementadas;
- Definição de objetos básicos, como o objeto do jogo e o mapa (cenário) do jogo.
- Detecção e gerenciamento de eventos gerados por teclado, *mouse*, *joysticks*, etc;
- Algoritmos para desenho dos objetos do jogo, podendo ser 2D, 3D, etc;
- Funcionalidades úteis a jogos 3D, como iluminação e transformações geométricas;
- Execução dos sons do jogo em resposta a eventos ocorridos;
- Implementação de algoritmos de IA para os objetos inteligentes dos jogos;
- Implementação de algoritmos para troca remota de dados, gerenciamento de sessões, sincronização das informações compartilhadas do jogo, etc;
- Implementação de algoritmos para modelagem física de objetos.

Note-se que, devido a alguns dos requisitos serem bastante independentes, como é o caso do processamento necessário à execução de jogos em rede e ao processamento envolvido em IA, um *framework* de jogos também pode ser visto como a composição de diversos *frameworks* com papéis bem definidos. [8]

Em conseqüência dessa grande variedade de requisitos dos *frameworks*, muitos procuram limitar-se a determinados estilos de jogos, como, por exemplo, para jogos 2D, 2½D, 3D, jogos de turno (Damas e Xadrez), etc. O *framework* wGEM, por exemplo, envolve apenas o desenvolvimento de jogos 2D.

4.2. Arquitetura

A arquitetura de um *framework* corresponde exatamente à definição dos seus módulos e a como é realizada a interação entre eles. Tais módulos são responsáveis por identificar e gerenciar todos os eventos que ocorrem ao longo do jogo, desde eventos gerados pelo usuário até aqueles causados por interação entre os objetos do jogo. Por isso, existem normalmente módulos para manipulação de objetos, renderização, sonorização, IA, rede e outros.

A qualidade da arquitetura é um dos aspectos mais importantes de um *framework*. Tal arquitetura torna-se crítica pelo fato de que um *framework* de jogos, em geral, contém um conjunto de classes abstratas que requerem, portanto, uma adaptação para produzir cada jogo. Assim, é de extrema importância que a arquitetura do *framework* seja clara e suficiente para que tal tarefa seja possível. Infelizmente, não há consenso sobre arquitetura de *frameworks* nem muitas publicações sobre esse tema, o que implica num grande esforço de análise e generalização de diversas soluções para compilar características comuns [14].

Uma vez definida a arquitetura do *framework*, os seus diversos módulos são então implementados. Os módulos responsáveis por atividades como identificação de entrada de dados são muito mais simples do que os responsáveis por transformações geométricas e renderização de objetos 3D. A implementação dos módulos, portanto, envolve a criação de algoritmos específicos para diversas tarefas. No módulo que representa o objeto do jogo, por exemplo, se necessita de algoritmos básicos de deslocamento espacial, bem como de detecção de colisão com outros objetos do jogo. Módulos que representem repositórios para objetos do jogo, por exemplo, requerem estruturas de dados eficientes em uso de memória e CPU.

Geralmente durante a fase de implementação há sempre uma longa etapa de otimização. As otimizações realizadas no *framework* são de extrema importância, pois o seu desempenho tem impacto direto nos todos os jogos construídos sobre o *framework*.

4.3. Editores de Cenários

O uso de um *framework* por si só não é suficiente para garantir um rápido desenvolvimento. Diversas ferramentas de apoio são úteis, em particular um editor de cenários adaptado para o *framework*. Este editor visa permitir de uma forma visual e simples a definição de estágios de um jogo. As seguintes funcionalidades são esperadas:

- Definir visualmente o mapa do jogo, incluindo altura, largura, textura, tiles, etc;
- Definir os objetos do jogo, incluindo atributos como velocidade e identidade visual;
- Definir o cenário, possivelmente utilizando recursos de *drag and drop*, permitindo que os objetos sejam facilmente adicionados, removidos e editados;
- Salvar e carregar cenários previamente desenvolvidos;
- Exportar os cenários em formato compatível com o motor em uso.

5. wGEM

Resultado de uma dissertação de mestrado do Centro de Informática da UFPE [8], o *framework* wGEM consiste numa série de adaptações das características apresentadas na seção 4 para as limitações de J2ME. A seguir serão apresentados os resultados desse processo.

5.1. Framework

Na seção 5.1 são listados os principais componentes e requisitos que acreditamos devam estar presentes em um *framework* para jogos 2D baseado em J2ME. Estes requisitos têm como base os jogos 2D de plataforma (como Sonic®, Super Mario®, etc.), que representam um dos estilos de jogos 2D de maior sucesso e que, ao mesmo tempo,

demandam grande quantidade de recursos do *framework* em uso. Sendo assim, é exatamente este o estilo de jogo mais adequado ao uso do wGEM.

Representação do Objeto do Jogo	Os objetos do jogo (tais como, carro, bola, etc.) precisam de uma representação oferecida pelo <i>framework</i> que contenha tanto as informações necessárias para o seu gerenciamento (como posição, velocidade, dimensão e representação gráfica), quanto para a execução do jogo (como detecção de colisão com outros objetos).
Representação do Mapa do Jogo	O <i>framework</i> deve prover a utilização de mapas ¹ para facilitar o desenvolvimento de jogos com cenários complexos. Recursos como <i>scrolling</i> ² e mapas baseados em texturas ou <i>tiles</i> ³ devem ser oferecidos para tornar simples a criação de jogos.
Gerenciamento de Objetos	Uma das partes mais críticas de um <i>framework</i> , envolvendo funções de inicialização, armazenamento e acesso aos objetos do jogo que ele gerencia.
Gerenciamento de Entrada	Encarregado de identificar eventos ocorridos no teclado do dispositivo e encaminhá-los para o módulo responsável pelo gerenciamento do jogo.
Gerenciamento Gráfico	Responsável pela fase de apresentação gráfica do ciclo do jogo, envolvendo a coordenação do processo de desenho do mapa e dos objetos do jogo.
Gerenciamento do Jogo	Sua principal função é implementar o ciclo do jogo, contando para isso com o apoio dos módulos de gerenciamento de objetos, do mapa do jogo, de entrada e gráfico.
Gerenciamento de Rede	Responsável por implementar as funcionalidades de <i>download</i> e <i>upload</i> de arquivos utilizando o protocolo HTTP suportado pelo perfil MIDP.
Gerenciamento de Aplicação	Encarregado de implementar as funcionalidades exigidas pelo J2ME às aplicações do perfil MIDP para que estas possam ser gerenciadas pela máquina virtual.
Integração com o Editor de Cenários	Deve oferecer as funcionalidades necessárias para a importação do arquivo produzido pelo editor de cenários.

Tabela 1: Componentes e funcionalidades para o *framework* wGEM

A arquitetura do wGEM, que define basicamente como os módulos da se comunicam, é apresentada na Figura 1.

¹ O mapa de um jogo 2D corresponde a um retângulo, geralmente maior do que a tela gráfica, sobre o qual os objetos do jogo estão localizados e se movem. Associado a cada mapa sempre existe uma janela visível, que é a região do mapa visível na tela gráfica.

² *Scrolling* representa a mudança de posição da janela visível do mapa.

³ Uma técnica comum para a representação gráfica do mapa do jogo é dividi-lo em pequenas partes e associar pequenas imagens (*tiles*) a cada parte.

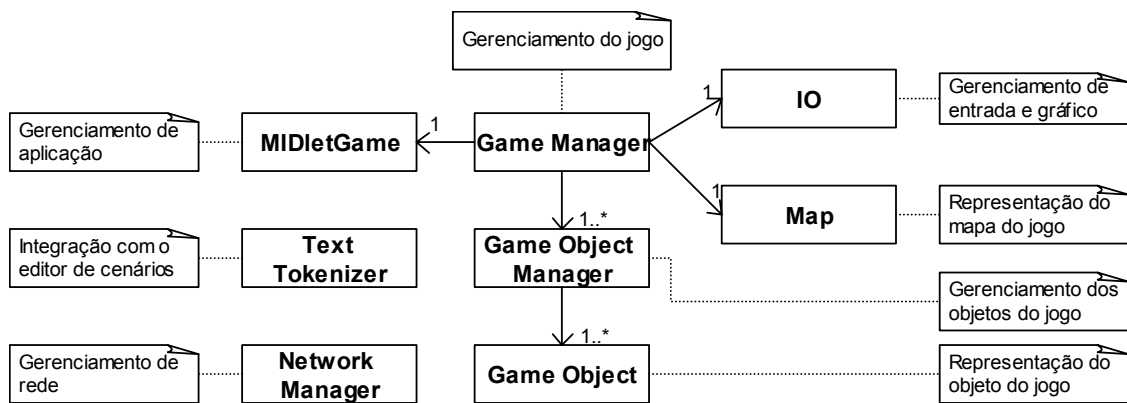


Figura 1: Arquitetura do *framework* wGEM

Esse modelo é fruto de nossa generalização de soluções para PCs e uma proposta de implementação e combinação dos componentes da para produzir o *framework* de jogos para o wGEM. Entre os principais pontos dos sistemas equivalentes considerados na construção desse modelo estão o mapeamento dos requisitos da para os módulos do *framework* [10] e a forma de cooperação entre eles [11].

5.2. Editor de Cenários

Baseado nos módulos identificados para o *framework* de jogos, os principais requisitos para o editor de cenários podem ser considerados. Na Tabela 2 são listadas as principais funcionalidades necessárias para tal editor. Recursos como a especificação de sons a partir do editor de cenários, apesar de ser um procedimento simples, não é considerado por não haver suporte a sons em J2ME.

Definição do Mapa	Através da interface deve ser possível especificar o tipo de mapa utilizado no jogo (baseado em textura, <i>tiles</i> , etc.) além de características como altura, largura, imagem de textura e imagem para <i>tiles</i> .
Definição dos Objetos	O editor deve prover um mecanismo onde os objetos do jogo a serem utilizados em cada cenário possam ser definidos pelo usuário. Todos os tipos de representação gráfica de objetos utilizados no <i>framework</i> devem poder ser especificados (baseados em imagens, formas geométricas, etc.). Os objetos definidos devem ser adicionados a uma paleta de objetos utilizada para a definição do cenário.
Definição do Cenário	A definição do cenário deve suportar recursos de <i>drag and drop</i> , permitindo que os objetos definidos e presentes na paleta de objetos sejam facilmente adicionados, removidos e deslocados dentro do cenário. Além disso, para cada objeto presente no cenário, características como a velocidade do objeto devem poder ser editadas.
Persistência de Cenários	O editor deve ser capaz de salvar e carregar cenários desenvolvidos, para facilitar a continuidade e a reutilização do trabalho.
Integração com o Framework	A exportação dos cenários desenvolvidos para um arquivo em formato compatível com o <i>framework</i> deve ser provida.

Tabela 2: Funcionalidades do editor de cenários do wGEM

6. Estudos de Caso

Para validar o wGEM foram implementados inicialmente dois jogos que podem ser executados no emulador J2ME produzido pela Sun [12]. O custo de execução do motor nos dois casos foi baixo. Nenhuma memória adicional é requerida durante o jogo, além daquela utilizada na inicialização dos cenários. Além disso, o desempenho foi muito satisfatório, atingindo taxas de 100 fps no emulador citado e 30 fps em dispositivos reais (Palms e celulares).

O primeiro jogo implementado foi o clássico BreakOut (ver Figura 2), onde o jogador tem o objetivo de destruir um conjunto de blocos utilizando uma bola controlada por uma raquete. O usuário deve controlar a raquete para evitar que a bola saia do campo de jogo e, ao mesmo tempo, tenta dar uma direção à bola que provoque o choque com algum bloco.

Os seguintes gerenciadores de objetos foram modelados para o BreakOut:

- Gerenciador da bola. Responsável por identificar colisão da bola com a raquete, os tijolos e a parede, assim como tomar as ações coerentes, como refletir a trajetória da bola, “avisar” a um tijolo que houve colisão, etc;
- Gerenciador da raquete, que inclui movimentar a raquete em função dos comandos do usuário;
- Gerenciador dos tijolos, capaz de inicializar todos os tijolos com suas devidas características e removê-los em caso de colisões.

O segundo jogo implementado foi denominado Ship (ver Figura 2), um jogo de nave no estilo RiverRaid, onde o jogador deve controlar uma nave por um campo contendo algumas naves inimigas, destruir o máximo de inimigos possível utilizando as armas da nave e coletar alguns itens de energia e armas especiais pelo caminho.

Os seguintes gerenciadores de objetos foram modelados para o Ship:

- Gerenciador da nave, responsável por identificar colisões dela com os inimigos, a borda da tela e os itens a serem coletados, assim como responder aos comandos do jogador (movimentação e tiro);
- Gerenciador dos inimigos, que inicializa os inimigos no momento previsto e administra seu ciclo de vida;
- Gerenciador dos tiros da nave do jogador, que permite a visualização dos tiros e que detecta colisão entre eles e os inimigos, tomando as providências devidas;
- Gerenciador das balas dos inimigos, idem ao caso anterior.

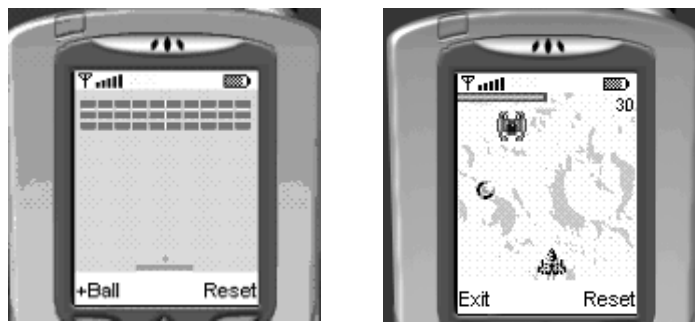


Figura 2: Jogos BreakOut e Ship, respectivamente, em execução no emulador

7. Conclusões

O surgimento de novas tecnologias como a J2ME tem atuado como um grande catalizador para a convergência digital ao tornar o desenvolvimento de aplicações para sistemas embarcados tão fácil e acessível ao público quanto no caso dos PCs. Por outro lado, a abertura desse mercado, aliado ao sucesso vivenciado pelo entretenimento digital na indústria de PCs, é um forte sinal do sucesso esperado da área de jogos para dispositivos móveis.

Nesse contexto, o wGEM é um esforço pioneiro que traz para esse novo mercado de dispositivos móveis uma base para o desenvolvimento de jogos usando o estado da arte em engenharia de software.

Nossos trabalhos futuros envolvem a implementação de objetos com comportamento inteligentes no wGEM (como perseguição e fuga de um dado objeto especificado) e a otimização no processo de detecção de colisão do gerenciador de objetos.

8. Referências

- [1] Java 2 Platform, Enterprise Edition (J2EE), URL: <http://java.sun.com/j2ee/>
- [2] Java 2 Platform, Standard Edition (J2SE), URL: <http://java.sun.com/j2se/>
- [3] Java2 Platform, Micro Edition (J2ME), URL: <http://java.sun.com/j2me/>
- [4] CDC and the CVM Virtual Machine, URL: <http://java.sun.com/products/cdc/>
- [5] CLDC and the K Virtual Machine (KVM), URL: <http://java.sun.com/products/cldc/>
- [6] Mobile Information Device Profile (MIDP), URL: <http://java.sun.com/products/midp/>
- [7] Foundation Profile, URL: <http://java.sun.com/products/foundation/>
- [8] Pessoa, C. wGEM: Um *Framework* de Desenvolvimento de Jogos para Dispositivos Móveis, Dissertação de Mestrado, Pernambuco, Novembro, 2001.
- [9] Feijó, B., Dreux, M., Ramalho, G., Battaiola, A., Pessoa, C., Desenvolvimento de Jogos em Computadores e Celulares, Florianópolis, SIBGRAPI 2001.
- [10] Fan, J. et al. Black Art of Java Game Programming, Waite Group Press, 1996
- [11] Fillion, D., Structure of Games, URL: <http://members.nbc.com/armagammon/articles/structure.htm>.
- [12] Java 2 Platform Micro Edition, Wireless Toolkit, URL: <http://www.java.sun.com/products/j2mewtoolkit/>
- [13] Centro de Estudos e Sistemas Avançados do Recife, URL: <http://www.cesar.org.br>
- [14] Madeira, C., Araújo, A., Macedo, H., Andrade, R., Cavalcanti, D., Ferraz, F. & Ramalho, G. NetMaze: um jogo de ação multimídia distribuído. *In: Simpósio Brasileiro de Sistemas Multimídia e Hipermídia (SBMIDIA'2000)*. pp. 129-139. Natal: SBC 2000
- [15] BREW Home Page, URL: <http://www.qualcomm.com/brew/>
- [16] JavaMobiles.com, URL: <http://www.javamobiles.com>