Um Motor para Jogos Cross-media

A. S. Silva & H. T. Macedo

Departamento de Computação, Universidade Federal de Sergipe, 49100-000, São Cristóvão-SE, Brasil

anderson.ufs@gmail.com, hendrik@ufs.br

(Recebido em 24 de março de 2009; aceito em 23 de setembro de 2009)

Este artigo apresenta uma proposta para o desenvolvimento de jogos cross-media: um novo estilo de jogo onde a principal característica é o compartilhamento de ambientes em diferentes plataformas. O objetivo do trabalho é permitir que o processo de construção do jogo passe por uma única fase de codificação independente do número de plataformas-alvo. Um motor adaptativo foi desenvolvido para validar a metodologia proposta e um estudo de caso também é apresentado com diferentes testes de desempenho.
Palavras-chave: jogos cross-media, motor adaptativo, multiplataforma

This paper proposes an innovative game development environment to support the development of cross-media games. A cross-media game is a new kind of game which main characteristic is the sharing of environments in different platforms. The main goal of the work is to enable the development of a game code regardless the target platform. An adaptive game engine has been developed in order to validate the proposal along with a case study. Key requirements to the development of a cross-media game are defined and the results of some experiments have shown the practicability of the proposal.
Keywords: cross-media games, adaptive engine, multiplatform

1. INTRODUÇÃO

Jogos para computador é um mercado em franca expansão e já há algum tempo tornou-se objeto de pesquisa no meio acadêmico. Esta é uma área muito extensa e em constante desenvolvimento. Linhas alternativas de investigação surgem a cada instante graças à criação e popularização de novas tecnologias que podem ser exploradas.

Os dispositivos móveis (e.g. celular, PDA - Personal Digital Assistent) e a criação de linguagens específicas (e.g. J2ME, BREW) formaram um terreno novo que atualmente é bastante explorado pelos desenvolvedores de jogos. A empresa especializada em jogos para dispositivos móveis Gameloft, por exemplo, além de desenvolver seus próprios jogos é contratada por outras empresas para desenvolver a versão móvel de um produto (e.g. Shrek the Third – The Mobile Game). A versão desse mesmo jogo para computadores de mesa foi desenvolvida por outra empresa, Electronic Arts, cuja especialidade são jogos para computadores e vídeo games. Não é novidade que o estado da arte de jogos de computador de mesa é bastante desenvolvido e, como ilustra o exemplo citado anteriormente, a área de jogos para dispositivos móveis também está seguindo o mesmo rumo. Os jogos cross-media surgiram como conseqüência dessa evolução. Esse novo gênero atualiza o conceito de jogos multiplataforma que até o momento é limitado aos sistemas operacionais e computadores de grande porte.

O jogo Hinterwars é um bom exemplo de jogo cross-media (figura 2). Os jogadores podem atuar na partida com as mesmas funcionalidades independentemente de qual plataforma suportada estiver utilizando. Além disso, o ambiente de jogo é o mesmo. Portabilidade e adaptação são as características que mais chamam atenção para quem vai desenvolver esse gênero de jogo.

Um projeto de jogo é dividido em diversas áreas. Em meados da década de 90 as funcionalidades eram modularizadas basicamente em som, gráficos 2D, fluxo de entrada e saída de dados e simulação [1]. Com o passar do tempo novas áreas foram agregadas ao projeto (e.g. Inteligência Artificial, detecção de colisão e física, layout de ambiente, entre outras), por isso, a complexidade aumentou bastante e fez com que aumentasse também a demanda por ferramentas que otimizassem a execução do projeto. Em [1] Blow faz um aprofundamento sobre desenvolvimento de jogos e mostra que a tarefa dos desenvolvedores é extremamente complexa.

Em um projeto de jogo 3D que siga o padrão do estado da arte atual, Blow sugere utilizar um Motor de Jogo como forma segura para conclusão do projeto. Alunos de graduação também podem utilizar essas ferramentas para gerar simulações de ambientes virtuais e testar diferentes abordagens de IA para jogos [2].

Figura 2: Screenshot do jogo Hinterwars.

Por se tratar de uma área relativamente nova e cheia de características particulares, o desenvolvimento de jogos cross-media ainda é feito de forma "artesanal". Não existem ferramentas para aumentar a produtividade nem metodologia de desenvolvimento. Seria interessante para o progresso dessa área se existisse uma ferramenta, de preferência open-source, que abstraísse os detalhes específicos de cada plataforma no nível de codificação. Essa ferramenta deveria permitir uma única etapa de codificação para se obter n versões para as n respectivas plataformas, considerando questões de desempenho, portabilidade e interface, já que as versões finais devem ser, ou aparentar ser, idênticas. A discrepância de poder de processamento e armazenamento, além da própria usabilidade dos dispositivos, faz com que as características citadas anteriormente se tornem o fator de maior risco em um projeto de jogo cross-media.

O desenvolvimento de jogos convencionais (e.g. Corrida, Simulador de Futebol, First Person Shooter, Rolling Player Game) tem o apoio de centenas de Motores de Jogos que vêm sendo construídos e consolidados há algum tempo. Essas ferramentas aumentam a produtividade da execução dos projetos. Para mostrar sua importância podemos citar o motor Quake3 e Unreal, que custam cerca de $450 mil por licença de máquina [1]. Por se tratar de uma área nova, os jogos cross-media não possuem muitas alternativas de Motor de Jogo [3].

O objetivo desse artigo é propor uma abordagem para desenvolvimento de jogos cross-media descrevendo as primeiras impressões do CORAGE – Code Onde Run Anywhere Game Engine, um conjunto de ferramentas que compõe um motor de jogo específico para jogos cross-media. Através de um estudo de caso, o artigo discute as características mais importantes que devem estar presentes nesse gênero de jogo à medida que apresenta sua solução.

A seção 2 faz uma breve introdução a motores de jogo e um aprofundamento sobre jogos cross-media. Na seção 3, o motor adaptativo CORAGE é detalhado. Um estudo de caso que ilustra a aplcação do motor CORAGE é apresentado na seção 4. Alguns experimentos e resultados são apresentados na seção 5 e, finalmente, as conclusões do trabalho são apresentadas na seção 6.

2. MOTOR DE JOGO

Motor de jogo é um conjunto de software reutilizável que permite a construção de jogos de forma mais simples e com menos riscos para os desenvolvedores. Seu principal objetivo é abstrair detalhes de baixo nível das funções que o compõe através de um conjunto de interfaces intuitivas para o usuário. De maneira geral, um motor de jogo simples é composto por:

- Motor gráfico – responsável pelo carregamento e renderização de imagens;

- Motor de som – responsável pelo carregamento e reprodução das músicas e efeitos especiais;
- Motor de física – responsável pela detecção de colisão (dependendo do motor, podem existir outros recursos como controle de aceleração e velocidade, gravidade e operações sobre vetor).

Cada um desses componentes pode estar presente em módulos diferentes ou em um único módulo, dependendo da complexidade das funcionalidades que o motor oferece e do gosto do desenvolvedor. Existe uma grande quantidade de motores disponíveis na Internet, alguns open-source, outros proprietários. Independentemente, esta área está bem desenvolvida. O RAGE engine [4], por exemplo, é um Motor de Jogo 3D multiplataforma, open-source, escrito em Java. Além do padrão, ele oferece recursos de comunicação peer-to-peer e máquinas de estados para implementação da Inteligência Artificial. Suas características possibilitam que ele seja comparado aos melhores motores de jogos proprietários.

Assim como no RAGE, o conceito de multiplataforma comumente observado nos jogos atuais é aplicado apenas no domínio dos Sistemas Operacionais. Em outras palavras, a maior parte dos títulos multiplataforma disponíveis no mercado não atingem os usuários de dispositivos móveis. Títulos como The Sims e Fifa Soccer, por exemplo, foram desenvolvidos inicialmente para computadores de mesa, aprovados pelos consumidores e depois ganharam uma versão para dispositivo móvel. Essas versões compartilham o mesmo tema, porém, não compartilham o mesmo ambiente de jogo.

O baixo poder de processamento e armazenamento que os dispositivos móveis mais populares apresentam é talvez o principal fator que afasta o interesse dos desenvolvedores por esse grande terreno que está se formando. Por outro lado, o rápido avanço tecnológico está provocando uma queda nos preços de dispositivos mais robustos e aumentando o número de possíveis consumidores. Em [5] Koivisto mostra que os jogos cross-media [6], jogos multiplataforma com acesso ao mesmo ambiente, serão muito mais comuns num futuro próximo.

Uma ferramenta para o desenvolvimento desse estilo de jogo pode ser construída a partir de um conjunto específico de componentes para cada plataforma e uma interface comum encarregada de fazer as chamadas específicas dependendo da escolha da plataforma-alvo. Outra possível abordagem é desenvolver um conjunto de componentes para as plataformas-alvo com interfaces similares (figura 3). Nesse caso pode ser difícil, ou mesmo impossível, construir a ferramenta utilizando linguagens de programação com características muito contrastantes. Por outro lado, pode ser simples se for selecionada uma boa linguagem. Dessa forma o desenvolvedor codifica o jogo para diferentes plataformas ao mesmo tempo.

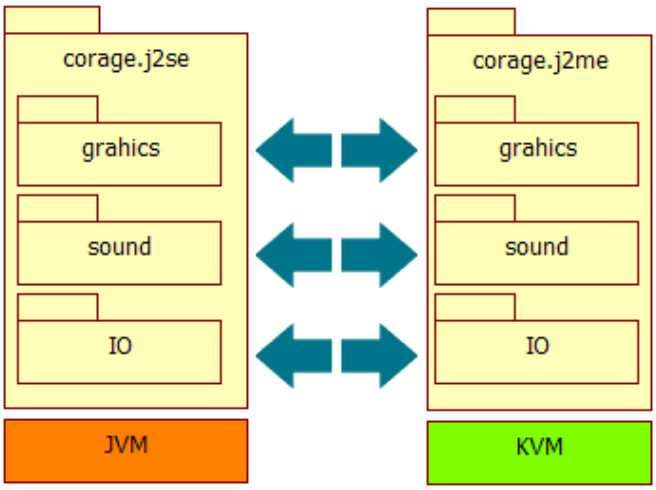

Figura 3 – Exemplo simplificado da abordagem proposta.

A linguagem Java é uma excelente escolha para construção desse tipo de motor. As versões J2ME (de Java to Mobile Edition) e J2SE (de Java to Standard Edition) podem ser utilizadas na

codificação dos componentes para dispositivos móveis e computadores de mesa, respectivamente. A sintaxe semelhante, a existência de IDEs poderosas e o estado da arte avançado dos motores de jogos para linguagem de programação Java são alguns fatores favoráveis para escolha dessa solução.

Um possível cenário ideal para jogos cross-media, que deve ser considerado no desenvolvimento do motor de jogo específico, deve respeitar os seguintes requisitos:

1. O desempenho do jogo não deve prejudicar nenhuma plataforma;
2. Os jogadores não devem ser beneficiados pela plataforma que está utilizando, ou seja, os jogadores devem ter o mesmo poder de atuação nas partidas;
3. A usabilidade do jogo não deve discriminar nenhuma plataforma.

Obedecer a esses requisitos é um desafio. O dispositivo móvel será por um bom tempo o fator limitante no desenvolvimento de jogos cross-media, basta observar o cenário atual dos jogos de PCs e de dispositivos móveis. A chave é buscar a interseção perfeita entre as tecnologias das plataformas e sempre que possível utilizar o que cada uma tem de melhor.

Um motor de jogo cross-media deve oferecer, além dos recursos tradicionais, um bom motor de conectividade para permitir a computação ubíqua e um motor de inteligência artificial. O perfil dos jogadores está cada vez mais exigente em relação aos agentes controlados pelo computador, por isso, inteligência artificial deve ter a mesma importância que o som e gráfico em um projeto de jogo. Em [7] Lent e Laird mostra que a partir da experiência no desenvolvimento de agentes para simulação de combate aéreo para DARPA (Defense Advanced Research Projects Agency) conseguiram aplicar alguns conceitos ao desenvolvimento de jogos. Nesse trabalho os autores descrevem a construção de um motor de inteligência artificial a partir de um motor de inferência e scripts contendo uma seqüência de ações para formar o comportamento de um agente. Existem motores de inferência desenvolvidos em diversas linguagens. O JEOPS [8], por exemplo, gera um motor de inferência para Java a partir de um arquivo contendo regras de produção. Essa ferramenta torna a modelagem de agentes inteligentes mais fácil e pode ser utilizada na construção de jogos.

A comunicação, por sua vez, é talvez o aspecto mais crítico em jogos cross-media. É essencial a comunicação entre os diferentes dispositivos para que haja interatividade entre os jogadores. As tecnologias sem fio Bluetooth e WiFi são comuns em dispositivos móveis e computadores de mesa, por isso, podem ser utilizadas como solução para essa integração. A ferramenta Marge, por exemplo, é uma API que facilita o desenvolvimento de aplicações sobre Bluetooth e pode ser utilizada tanto em J2ME como em J2SE, por isso é uma excelente ferramenta para ser integrada aos Motores de jogos cross-media.

Através de uma pesquisa foi constatado que não existem ferramentas open-source específicas para construção de jogos cross-media. O JGame foi a solução mais próxima. Com uma única fase de codificação e poucas alterações, esse motor consegue gerar jogos em Applet e J2ME. Por outro lado, não disponibiliza motor de conectividade e inteligência artificial. A seguir é apresentada uma descrição do CORAGE, um conjunto de soluções que segue os princípios citados anteriormente.

3. O MOTOR ADAPTATIVO CORAGE

O CORAGE foi construído seguindo a proposta apresentada no artigo. Utilizando a linguagem Java nas versões J2ME e J2SE foram montadas soluções para motor gráfico, motor de IA e motor de comunicação, com interfaces semelhantes.

O motor gráfico é formado pela Game API (pertencente ao pacote javax.lcdui.microedition) para J2ME e sua adaptação para J2SE (pertencente ao pacote corage.graphics). Essa API é composta por um conjunto de classes que facilita o desenvolvimento de jogos para dispositivos móveis. Através de uma interface simples, ela fornece recursos como; double buffering, detecção de colisão, sprites animados e tiledmaps. O padrão para montar um bom jogo 2D.

O processo de adaptação foi feito sem muitas complicações. A principal diferença é a presença da classe Image no pacote graphics, responsável pela criação de diferentes tipos de imagens. Em J2SE podemos criar três tipos de imagens:

1. Opaque: Não existe transparência;
2. Bitmask: Os pixels da imagem ou são transparentes ou não;
3. Translucent: Cada pixel da imagem tem seu nível de transparência (canal Alpha).

Cada um desses tipos implica de maneira diferente no desempenho do jogo (veja na seção 5 o gráfico comparativo). Outra característica importante do pacote "graphics" é a utilização do FSEM (de Full Screen Exclusive Mode) como padrão. Simulações construídas nesse modo têm um desempenho melhor quando comparados com o modo janela [9]. Esse ganho por parte da plataforma J2SE pode não ser tão importante quando o hardware do dispositivo móvel for muito limitado. Porém, existem configurações de dispositivos móveis impressionantes. Por exemplo, o Sony Playstation Portable tem um processador de 333MHz e 32Mb de memória e IEEE802.11 b (Wifi), uma boa configuração para rodar jogos cross-media. Oficialmente o PSP não vem com suporte a Java, mas essa configuração vai se tornar comum em outros dispositivos.

O motor de IA é composto por um gerador de motor de inferência para cada plataforma. Esse motor é gerado a partir de uma base de regras formada por regras de produção. Cada tipo de variável declarada na base de regras precisa estar presente na memória de trabalho do motor de inferência, em nosso caso, os agentes autônomos e elementos que interagem com eles. A cada iteração do loop do jogo o motor pesquisa em sua memória de trabalho na tentativa de casar fatos com a base de regras (veja no código 1 um exemplo de regra codificada no JEOPS) até que nenhuma regra possa ser disparada.

Código 1 – Exemplo de regra de produção no JEOPS

```
Rule attack {
declarations //declaração de variáveis
  Human h; Monster m;
conditions //apenas sentenças booleanas
  h.see(m);
  !h.disarmed();
actions
    h.attack(m);}
```

A melhor solução encontrada para o CORAGE foi o conjunto JEOPS e KEOPS [10]. O primeiro foi desenvolvido para plataforma J2SE e o segundo teve como objetivo portar o JEOPS para J2ME. Essa solução foi escolhida para o motor de IA porque são excelentes ferramentas e permitem a obtenção de um motor de inferência para cada plataforma a partir da mesma base de regras.

O motor de comunicação foi estruturado para realizar testes com a tecnologia Bluetooth. No processo de codificação do estudo de caso foi observado que utilizar o JSR82 (API Java para Bluetooth) diretamente era inviável dado sua complexidade. Foi necessário um estudo mais aprofundado sobre a pilha de protocolo Bluetooth e concluímos que seus detalhes precisavam ser abstraídos. Procurando esconder esses detalhes de baixo nível encontramos o framework Marge. Ele funciona acima do JSR82 sendo portanto independente da máquina virtual que está executando; basta conseguir a implementação da JSR82 para J2SE e J2ME. O Wireless Toolkit, ferramenta para desenvolvimento de aplicações em J2ME, já vem com o JSR82 embutido em sua distribuição e existem diversas alternativas para J2SE (e.g Electric Blue, Avetana Bluetooth). Para formar o motor de comunicação optamos pelo BlueCove porque se mostrou mais estável. [11] traz mais detalhes sobre jogos ubíquos com Bluetooth. A figura 4 mostra como o CORAGE está estruturado atualmente:

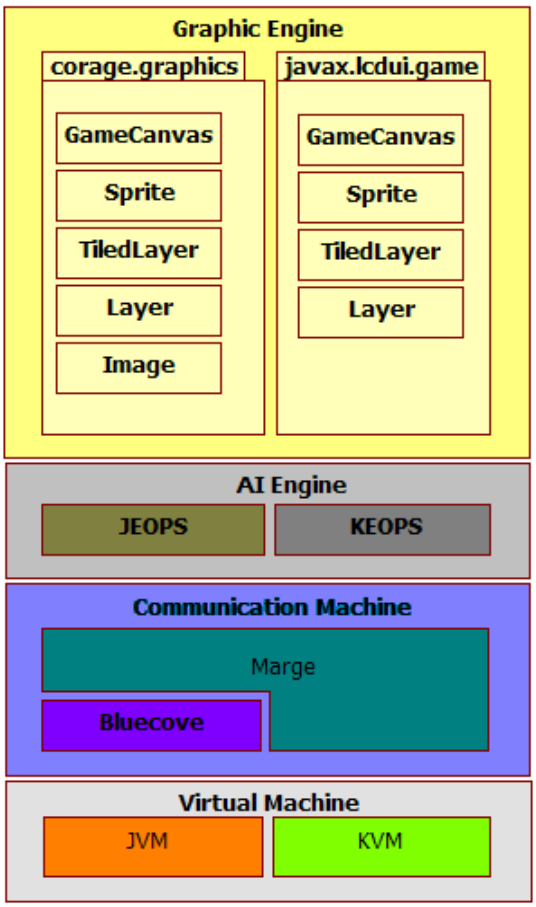

Figura 4 – Estrutura do CORAGE

A próxima seção descreve um estudo de caso desenvolvido com o CORAGE: um jogo cross-media dinâmico que pode ser executado em qualquer combinação de plataforma que tenha suporte a J2SE ou J2ME (e.g. PC, PDA, Celular).

4. ESTUDO DE CASO

O jogo trata de uma simulação baseada no jogo clássico PacMan. O objetivo é tentar comer todos os pontos disponíveis no labirinto sem entrar em contato com os fantasmas. Nas simulações o PacMan é controlado por um motor de inferência e os fantasmas realizam apenas movimentos aleatórios (no clássico, os fantasmas são agentes autônomos controlados por máquina de estados).

O processo de codificação foi realizado apenas uma vez e as poucas adaptações necessárias podem ser facilmente automatizadas. Na plataforma J2SE foi necessário criar o método “main” para que o jogo pudesse ser inicializado. Em J2ME todas as aplicações devem estender a classe Midlet que contém um conjunto de métodos para controlar seu ciclo de vida, portanto, o jogo precisou ser instanciado em uma classe a parte (veja na figura 5 a UML do jogo).

O processo de codificação não implicou com nenhum requisito do cenário ideal proposto na seção 2. Foi possível desenvolver o jogo compatível com as duas plataformas oferecendo as mesmas funcionalidades, ou seja, os jogadores podem atuar no jogo da mesma maneira independente da plataforma. Apesar de o jogo ser modelado em um ambiente dinâmico, foi possível realizar as simulações com velocidade confortável para as duas plataformas. No decorrer das simulações foram notados alguns movimentos inválidos decorrente da falta de sincronizmo. Esse problema não foi tratado porque é complexo e foge do escopo do trabalho.

A maior dificuldade encontrada no processo de adaptação diz respeito à usabilidade do jogo. Existem dispositivos que o tamanho da tela não é suficiente para caber os gráficos do jogo.

Nesses casos uma janela com o centro nas coordenadas do jogador vai se deslocando quando necessário. Essa abordagem prejudica o jogador limitando o acesso ao ambiente e pode variar de acordo com a configuração do dispositivo.

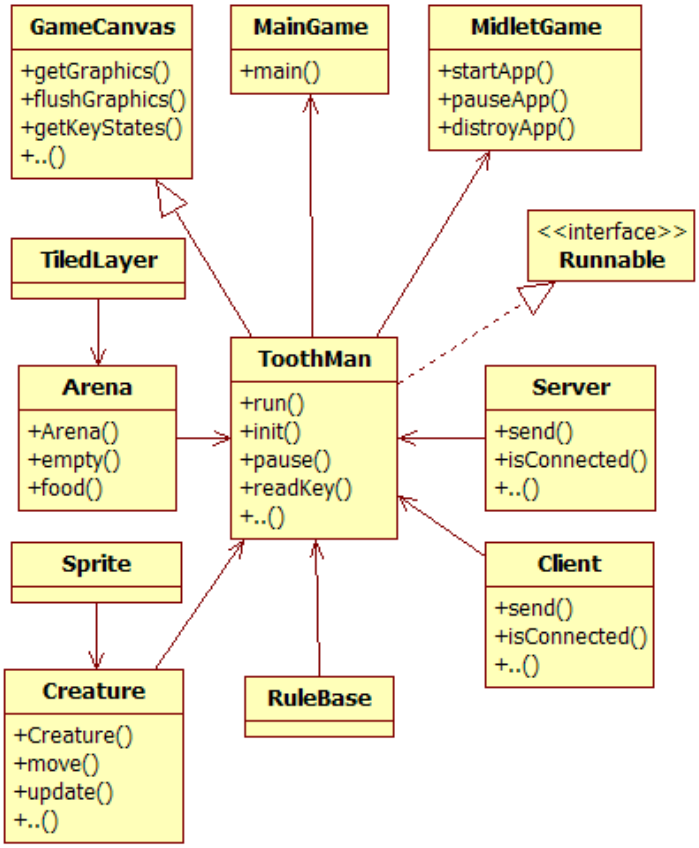

Figura 5: UML do jogo ToothMan

A interface do jogo é similar, foi necessário apenas redimensionar os gráficos da plataforma J2SE para J2ME (veja figura 6). Não é aconselhável fazer esse procedimento no sentido contrário porque os gráficos podem ficar distorcidos. Na próxima seção apresentamos as primeiras impressões do jogo construído com o CORAGE.

5. EXPERIMENTOS E RESULTADOS

A medida de desempenho utilizada nos testes realizados é chamada de FPS (de Frames per Second). Um frame representa uma passagem completa pelo loop do jogo (execução das atualizações e renderização). O limite inferior aceitável de FPS é determinado pelo olho humano. Nós podemos perceber a transição de frames abaixo de 10 a 50FPS, logo, as simulações devem estar executando acima dessa faixa. O limite superior é dado pela freqüência de atualização do monitor, um valor padrão para monitores de LCD é 60hertz que pode ser traduzido em 60FPS. Por outro lado, os dispositivos móveis apresentam uma variedade muito grande de configuração, por isso, não tem como definir um limite superior padrão.

O gráfico do jogo é composto por um grid de 25x27 posições preenchidas com imagens de 16x16 pixels (veja figura 6).

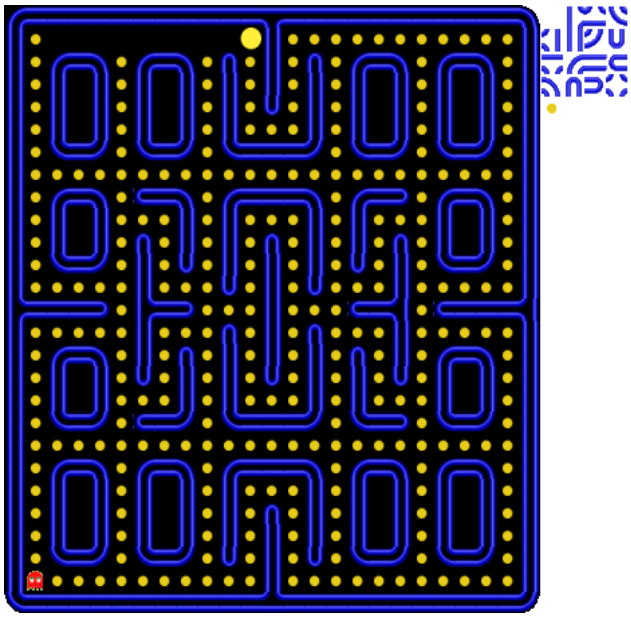

Figura 6 – Gráfico utilizado nos testes

As imagens utilizadas na versão para J2ME foram redimensionadas para metade do tamanho original visando tornar o aspecto do jogo confortável e compatível com as telas relativamente pequenas.

A seguir serão apresentados os resultados de alguns testes relevantes na criação do estudo de caso.

5.1 Modo Gráfico

Esse teste mostra a diferença de desempenho das simulações nos modos janela e FSEM (veja tabela 1). A resolução utilizada foi 800x600 pixels para os dois modos.

Tabela 1-Comparação de Modo Gráfico

FPS Requerido	50	80	100
Windowed	33	40	63
FSEM	50	64	64

Como era esperado, o modo FSEM obteve um desempenho superior ao modo janela. Por isso, o padrão adotado no motor gráfico é o FSEM, mas outros modos podem ser facilmente adaptados ao engine.

A próxima tabela faz uma comparação de desempenho entre os diferentes tipos de imagens existentes em J2SE (veja tabela 2).

Tabela 2 – Comparação entre os tipos de imagens

FPS Requerido		50	80	100
Windowed	Opaque	44	59	62
	BitMask	44	56	62
	Translucent	32	40	53
FSEM	Opaque	50	64	64
	BitMask	50	64	64
	Translucent	50	64	64

Com esses resultados constatamos que independentemente do tipo de imagem utilizada o modo FSEM não teve seu desempenho afetado (com o hardware utilizado nos testes), o mesmo não ocorreu com o modo janela. As imagens translucent são as que mais afetam o desempenho da simulação. O CORAGE cria esse tipo de imagem apenas quando é necessário.

5.2 Desempenho em Dispositivos Móveis

A figura 7 não tem como objetivo indicar qual foi o melhor dispositivo nos testes. Ela serve para confirmar a viabilidade do desenvolvimento de jogos cross-media no quesito desempenho em dispositivos móveis. Esses dispositivos apresentam diferentes resoluções de tela. Em alguns casos todo o tiled map pôde ser renderizado em outros apenas uma parte dele. Nesse caso, uma janela com centro no jogador foi definida para que pudesse "deslizar" sobre o mapa. O redimensionamento das imagens só faz sentido até certo ponto: em resoluções muito pequenas o gráfico do jogo pode ser tornar incompreensível.

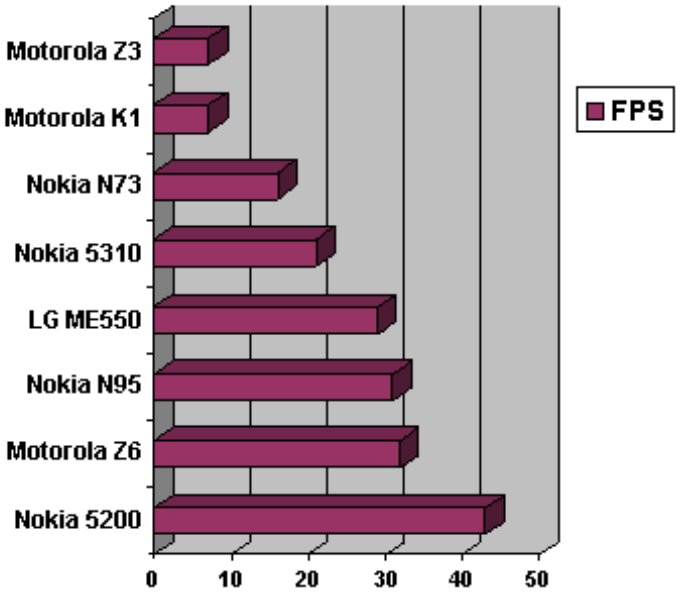

Figura 7 – Gráfico de desdemepho

Apesar de ter atingido o maior valor de FPS, o Nokia 5200 tem a menor resolução de tela utilizada nesse teste, que por sua vez significa uma menor exigência de hardware. A simulação foi visualmente confortável nos dispositivos que conseguiram ultrapassar 20FPS. Abaixo disso fica clara a transição entre os frames.

5.3 Tempo de Atualização

Esse teste foi realizado para ilustrar o problema da falta de sincronismo em jogos com ambientes dinâmicos construídos com Bluetooth (veja figura 8).

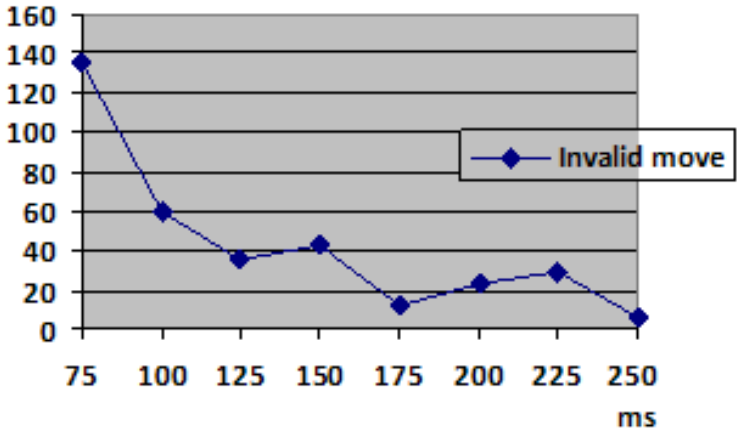

Figura 8 – Gráfico comparativo de movimentos inválidos e período de atualização

Quando o período do loop do jogo se aproxima de um valor aceitável, o número de movimentos inválidos aumenta drasticamente. O período de 75ms resulta em aproximadamente 13FPS e cerca de 130 movimentos inválidos por minuto. Para jogos com ambientes dinâmicos é impossível realizar simulações com esse problema. Por outro lado, jogos como o Hinterwars pode ser facilmente desenvolvido com Bluetooth sem que esse problema afete o desempenho do jogo.

6. CONCLUSÃO

O motor adaptativo CORAGE correspondeu a todas as expectativas. A partir de uma única codificação, foi possível gerar, com pequenas adaptações, um jogo cross-media para as plataformas desejadas. Apesar da diferença de desempenho observada nos testes entre as duas plataformas, o limite imposto pelos dispositivos móveis é aceitável para construção de jogos 2D. O experimento serviu para mostrar que a abordagem proposta é relevante e que essa área, apesar de complexa, já pode progredir e se tornar mais usual.

A prioridade no momento é resolver o problema de sincronismo para jogos dinâmicos com Bluetooth e propor uma solução para IEEE802.11 (wifi). Outro problema que está sendo investigado diz respeito á adaptação automática da interface do jogo para diferentes plataformas. Já existem alguns trabalhos nessa área. Em [12] é proposto um serviço de adaptação de interfaces para jogos multiusuários multiplataforma, porém, esse trabalho não condiz com alguns requisitos de jogos cross-media.

AGRADECIMENTOS

Os autores agradecem ao apoio do programa institucional PICVOL/UFS.

---

1. BLOW, J. Game development: Harder than you think. ACM Press Queue, 1(10) (2004).
2. PONSEN, M., MUNOZ-AVILA, H., SPRONCK, P., and AHA, D. Automatically Generating Game Tactics via Evolutionary Learning. *AI Magazine*, 27(3), AAAI Press, Madison, WI, 75-84 (2005).
3. BISHOP, L., EBERLY, D., WHITTED, T., FINCH, M., SHANTZ, M., Designing a PC Game Engine, *IEEE Computer Graphics and Applications.* v.18 n.1, p.46-53 (1998).
4. McCULLOCH, L., HOFMAN, A., TULIP, J., ANTOLOVICH, M., RAGE – A Multiplatform Game Engine. *ACM International Conference Proceeding Series* 123: 23–25 (2005).
5. KOIVISTO, E., Mobile Games 2010. *ACM International Conference Proceeding Series* Vol. 223 (2006).
6. KOIVISTO, E., WENNINGER, C.: *Enhancing Player Experience in MMORPGs with Mobile Features.* In Proceedings of DIGRA conference, Vancouver, Canada (2005).
7. LAIRD, J. and van LENT, M. *Developing an Artificial Intelligence Engine*. In Proceedings of the Game Developers' Conference, San Jose, CA, 577-588 (1999).
8. FILHO, C., RAMALHO, G. *JEOPS - The Java Embedded Object Production System.* Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI: Advances in Artificial Intelligence, p.53-62, November 19-22, (2000).
9. INGLES, B. *The Future of Java Game Development*. Proceedings of the 44th annual southeast regional conference (2006).
10. ALBUQUERQUE, R.; GUEDES, P.; FIGUEIRA, C.; ROBIN, J. and RAMALHO, G. *Embedding J2ME-Based Inference Engine in Handheld Devices: The KEOPS Case Study*. In Proceedings of the Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices. Italy (2002).
11. CHEOK, A., FONG, S., GOH, K., YANG, X., LIU, W. FARZBIZ, F. *Human Pacman: a sensing-based mobile entertainment system with ubiquitous computing and tangible interaction*, Proceedings of the 2nd workshop on Network and system support for games, p.106-117, May 22-23, Redwood City, California (2003).
12. TRINTA, F., FERRAZ, C. and RAMALHO, G. *Middleware Services for Pervasive Multiplatform Networked Games*. In Proceedings of ACM Workshops on Network and System Support for Games'2006. ACM Press (2006).