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

INCORPORANDO COMPORTAMENTOS DE MOVIMENTAÇÃO E ASPECTOS DE MODELAGEM FÍSICA AO WGEM

TRABALHO DE GRADUAÇÃO

Aluno: Börje Felipe Fernades Karlsson (bffk@cin.ufpe.br)
Orientador: Geber Ramalho (glr@cin.ufpe.br)

Abril de 2002

Resumo

Como pode ser visto pela quantidade e diversidade dos jogos atuais, esta é uma área em franca expansão, movimentando anualmente milhões de dólares no mundo todo. Além do tamanho do mercado jogos exigem state of the art. Um grande fator para esse sucesso, além do enorme interesse do público, é a diversidade de equipamentos que vem sendo criada continuamente ao longo desse tempo, desde PCs, videogames até uma que merece uma especial atenção, os dispositivos móveis (em especial celulares).

Tal interesse especial vem do fato de as novas gerações desses equipamentos disporem de uma capacidade de processamento bastante razoável e o do grande crescimento do mercado *wireless*. Além disso a disponibilidade de diversos ambientes de desenvolvimento [4][5][11] torna o desenvolvimento de jogos para celulares não só viável, mas também uma área bastante promissora. Pensando nesse mercado, e no fato da tecnologia J2ME ter sido apoiada por um consórcios de grandes empresas, não ficando assim restrito a poucas arquiteturas, foi criado um *framework* para facilitar a criação de jogos para celulares e proteger o desenvolvedor das características específicas das plataformas e da implementação das rotinas básicas da maioria dos jogos deixando que este se foque nas características do jogo, este *framework* foi denominado wGEM (**Wireless Game Engine for Mobile Devices**)[20].

O desenvolvimento do *framework* wGEM foi focado principalmente na arquitetura básica do motor e no gerenciador gráfico. Para serem criados jogos com mais jogabilidade e maior variedade de caracteristicas é necessário um melhor suporte a mais áreas.

Neste trabalho optamos por tratar da modelagem física e a partir desta modelagem implementar uma camada de IA para modelagem de comportamento de movimentação de modo a prover uma maneira fácil de utilizar esses recursos na criação de jogos para tentar alcançar o objetivo citado anteriormente.

Palavras-Chaves: Jogos de Computador, Programação para Celulares, “Steering Behaviors”, Modelagem Física e Engenharia de Software.

Agradecimentos

Ao meu orientador Geber Ramalho.

A todos os colegas que me ajudaram direta ou indiretamente para a realização desse trabalho, em especial a Paulo Abadie por discutir comigo vários aspectos desse trabalho e a Aydano por sua ajuda inicial com os comportamentos.

Índice

1 Wireless Games 7
2 Java 2 Micro Edition – J2ME 8
2.1 Mobile Information Device Profile – MIDP 10
2.2 MIDP x Desenvolvimento de Jogos 11
3 wGEM 13
3.1 Pendências do wGEM 14
4 Extensão proposta ao wGEM 15
4.1 Objetivos 15
4.2 Motivação 15
4.3 Estado da arte 16
5 Implementação e Resultados 22
5.1 Modelagem física 22
5.2 Comportamentos de movimentação 22
5.3 Estudos de caso 23
6 Conclusões 24
7 Bibliografia 26

Introdução

Nas últimas décadas, vem havendo um grande crescimento no mercado e complexidade dos jogos eletrônicos. Esse crescimento acabou por acarretar em uma grande mudança na metodologia de desenvolvimento [17] de jogos de modo que hoje em dia a indústria de jogos tem dado bastante importância aos benefícios que a Engenharia de Software pode prover.

Em se tratando de jogos, uma boa arquitetura de software pode trazer enormes benefícios. Principalmente devido ao fato de que uma grande parte do ciclo de desenvolvimento de jogos necessita de módulos que são comuns a diversos jogos. Sendo assim, a reutilização de tais módulos proporciona não só ganho de tempo, mas também de qualidade, uma vez que uma maior parte do ciclo de desenvolvimento poderá ser dedicado ao jogo em si, e não mais a detalhes técnicos sobre tarefas secundárias e características peculiares à plataforma de execução do jogo e esses módulos já foram bem testados.

Outro ponto importante é o enorme crescimento do mercado e poder de processamento de dispositivos móveis, com esse crescimento diversas empresas passaram a oferecer a disponibilidade de diversos ambientes de desenvolvimento para seus dispositivos. Uma dessas propostas e que mereceu uma atenção especial devido ao fato de ser suportada por diversas empresas e ser multi-plataforma é a J2ME.

Essa API, que será bem detalhada mais adiante, foi resultado de uma simplificação seguida de especialização da linguagem Java usada em PCs [12], com o objetivo de usá-la também em dispositivos bastante simples (tais como celulares, *pagers*, carros, etc.). Esse processo resultou na divisão de Java em três edições, dais quais Java 2 Micro Edition - J2ME [11] é voltada para dispositivos com poucos recursos. Além disso, foram definidas extensões (chamadas de configurações e *profiles*) de J2ME específicas para certas categorias de dispositivos, dos quais *Mobile Information Device Profile* – MIDP [7] é o *profile* voltado para celulares e *pagers*.

Essa situação levou então a criação de um *framework* para o desenvolvimento de jogos para dispositivos móveis chamado wGEM que oferece para o desenvolvedor de jogos um ambiente onde este não precisa se preocupar com as peculiaridades da plataforma e com implementações básicas em jogos, podendo focar o tempo de desenvolvimento na idéia e na jogabilidade do jogo.

Com o crescimento do poder gráfico e de processamento dos dispositivos móveis, vem surgindo espaço para jogos cada vez mais complexos e para tomar partido desse progresso é

necessário continuar extendendo o wGEM acrescentando novas funcionalidades necessárias ao desenvolvimento de jogos.

Diante deste cenário, o presente trabalho tem alguns objetivos:

(i) Estudar "modelagem física" num modelo simples e matematicamente "leve" não necessariamente "realista". Esse estudo é de grande importância e um pouco difícil, uma vez que tais arquiteturas não são bem documentadas, além de as arquiteturas de desenvolvimento existentes são concentradas no gerenciamento gráfico;

(ii) A partir desse estudo propor um modelo simples de modelagem física que dê suporte a se criar jogos mais complexos e que também sirva como uma camada de apoio para funcionalidades ainda mais complexas, mas compatível com os recursos disponíveis nos dispositivos móveis atuais.

(iii) Implementar com essa base comportamentos de movimentação (*steering behaviors*) assim como demonstrações da utilização desses.

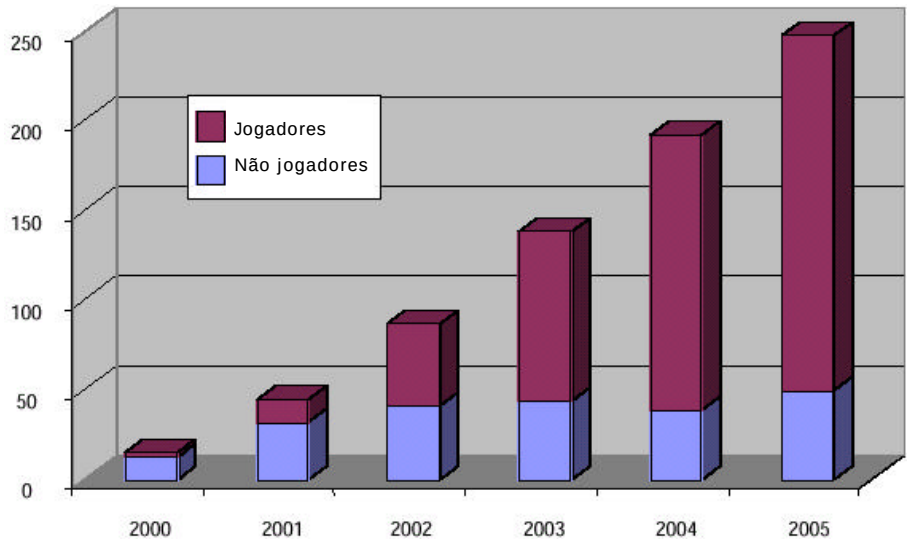
1 Wireless Games

Jogos móveis não são exatamente novidade, eles existem há décadas, primeiramente na forma de dispositivos especializados como o Gameboy da Nintendo e seus predescessores, e mais recentemente embutidos em muitos telefones celulares e PDAs.

Assim que os celulares passaram a ter telas razoavelmente largas e certo poder de processamento, os fabricantes passaram a vendê-los já com jogos pré-carregados (Snake e Memória, por exemplo). Mas estes eram simples e embora divertidos, não se comparam a gama de jogos encontrada hoje em dia.

Pensando no mercado de dispositivos móveis e no tamanho do mercado de jogos, várias empresas passaram a tentar prover APIs ou SDKs (BREW[4], J2ME[11], etc.) para o desenvolvimento de aplicações para seus dispositivos, sendo algumas destas com características claramente voltadas para jogos.

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 [18], outra pesquisa estima que haverão cerca de 200 milhões de jogadores em dispositivos móveis em 2005 [19] o sem dúvida representa um grande mercado para jogos nesses dispositivos.



Milhões de usuários de telefonia celular

2 Java 2 Micro Edition – J2ME

Desde a criação da linguagem Java esta já tinha como alvo dispositivos que não fossem computadores e ter a capacidade de execução em vários ambientes diferentes.
Tendo isso em vista, Java foi criada como uma plataforma pequena e simples. Porém, diante do seu sucesso, a cada nova versão da plataforma novos recursos são adicionados, tais como, Swing, Java 2D, Java 3D, etc. Conseqüentemente, a plataforma vem se tornando cada vez maior e exigindo mais memória e capacidade de processamento. Fica claro que essa situação contrasta com um dos principais objetivos de Java que é a portabilidade, uma vez que nem todos os dispositivos que suportem a execução de programas Java suportam tamanha demanda de recursos de processamento ou não possuem o recurso físico necessário.

Diante dessa situação, juntamente com o constante avanço de dispositivos tais como, celulares, *pagers* e *PDA's*, em 1999 a Sun dividiu a plataforma em três edições, cada uma atendendo a uma diferente categoria de dispositivos. Foram elas: Java 2 Enterprise Edition – J2EE, destinada a sistemas comerciais de grande porte onde tamanho e velocidade de processamento são providos pelo poder das máquinas [13]; Java 2 Standard Edition – J2SE, voltada para o mercado de computadores domésticos, no qual Java já é um grande sucesso hoje em dia [12] e Java 2 Micro Edition – J2ME, voltada para dispositivos com capacidade limitada de processamento e memória [11].

Além dessa divisão, a plataforma J2ME também foi desenvolvida de forma bastante modular, de forma que para cada categoria de dispositivo que utiliza J2ME exista uma especialização da plataforma específica (por exemplo, as rotinas de gerenciamento gráfico de um *pager* serão muito mais simples que as exigidas por um celular ou *PDA*). Sendo assim, seria cometer um grave erro pensar em aplicar a mesma plataforma (que teria que ser complexa) a essas duas categorias de dispositivos.

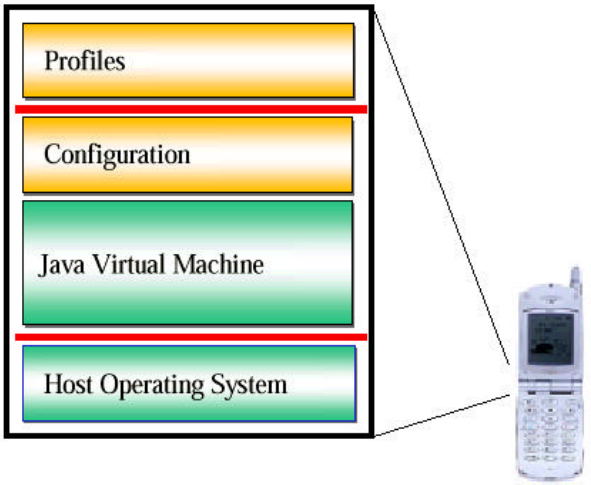
A técnica usada para proporcionar essa modularidade foi conseguida através de um modelo de três camadas construído sobre o sistema operacional presente em cada dispositivo: [15]

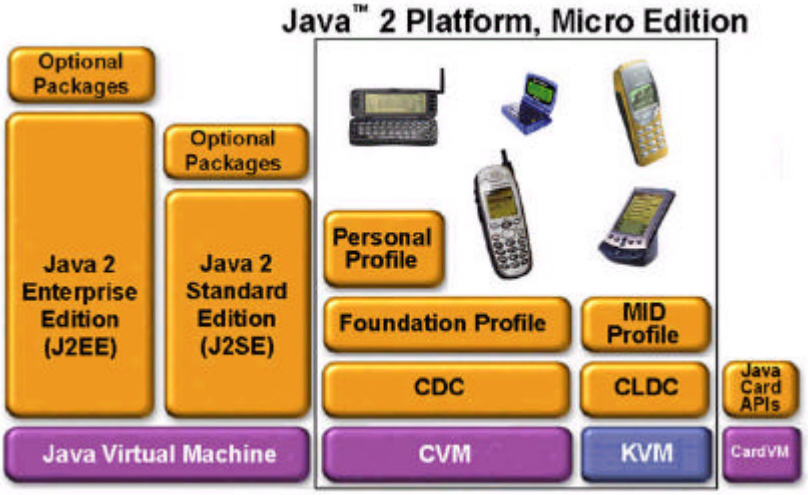
- *Java Virtual Machine (JVM) Layer*. Uma implementação da máquina virtual Java para cada dispositivo. Essa JVM é bem simples e proverá funcionalidades iguais para todos os dispositivos que usem J2ME. Ela é capaz apenas de executar pouquíssimas tarefas, pois o objetivo é que os recursos especiais suportados por cada categoria dispositivos sejam providos pelas configurações e *profiles* que irão especializar a JVM.

- *Configuration Layer*. Define uma plataforma mínima que é suportada por uma categoria particular de dispositivos que representa um segmento “horizontal” de mercado. Em outras palavras, uma configuração define o “menor denominador comum” dos recursos da plataforma Java que serão suportados em todos os dispositivos pertencentes a uma determinada categoria. Atualmente existem duas configurações, uma para dispositivos fixos e com boas capacidades de energia e conexão (Connected Device Configuration - CDC) [21] e outras para dispositivos móveis e mais simples (Connected, Limited Device Configuration - CLDC) [22].

- *Profile Layer*. Camada mais próxima para os usuários e desenvolvedores de aplicações. Ela define a API disponível em uma particular família de dispositivos que representa um segmento “vertical” de mercado (tais como celulares, televisores, etc). Assim, cada *profile* é implementado sobre uma determinada configuração para torná-la especializada para uma parte do mercado “horizontal” que ela atinge. Aplicações são desenvolvidas para um determinado *profile* com a garantia que será portável para qualquer dispositivo que suporte tal *profile*. Atualmente existe um *profile* definido que abrange celulares e *pagers,* Mobile Information Device Profile – MIDP [7].

Esses modelos em camadas são ilustrados nas figuras abaixo





2.1 Mobile Information Device Profile – MIDP

O wGEM foi implementado sobre o *profile* MIDP. Neste *profile,* qualquer desenvolvedor de aplicações pode modelar um sistema e ter a garantia de que o mesmo será executado pelos dispositivos que o implementam, independente do fabricante. A idéia é que isso ocorra da mesma maneira que aplicações são desenvolvidas para um determinado sistema operacional sem preocupação sobre o fabricante do computador que irá executá-lo.

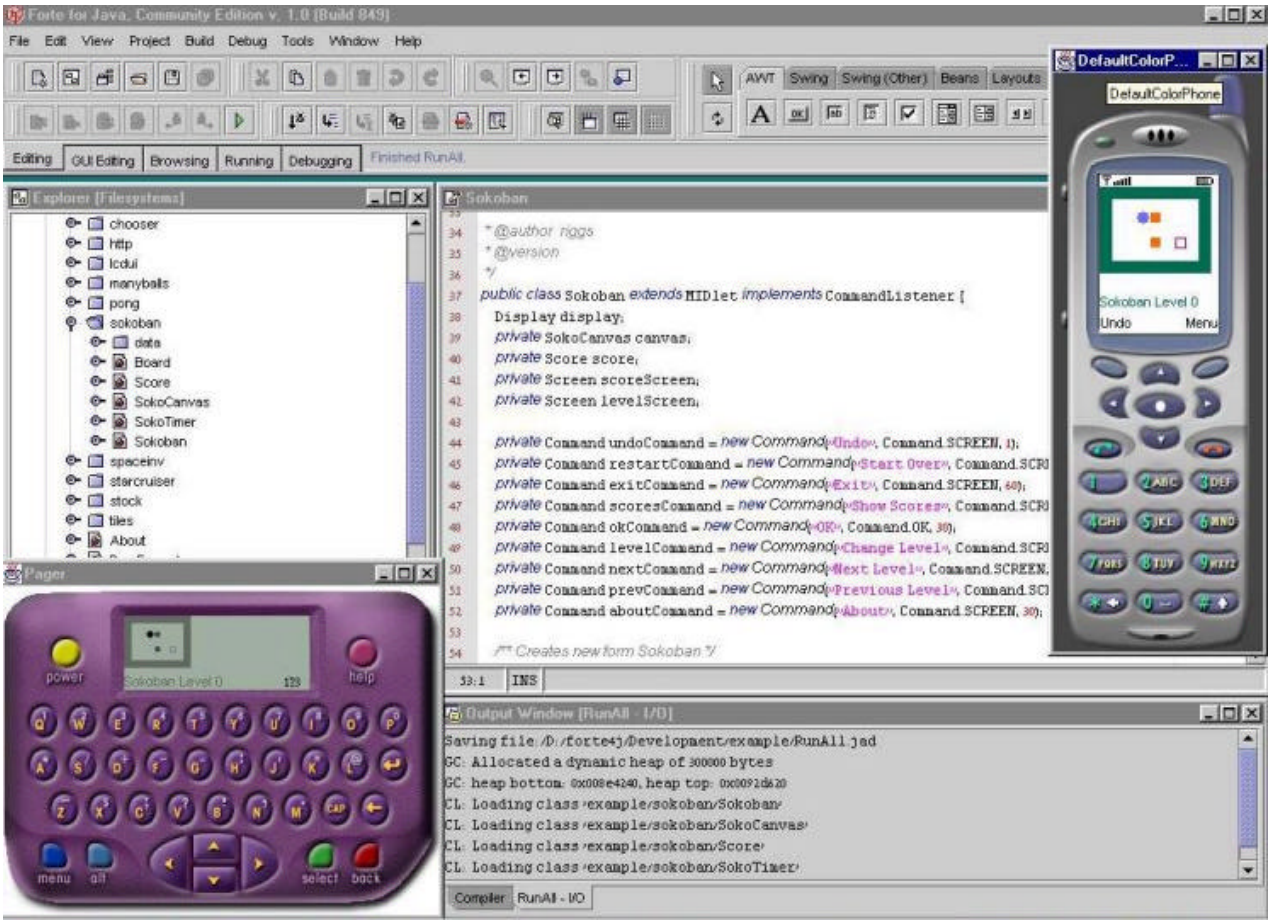
O MIDP estende a API básica de CLDC definindo basicamente um conjunto de classes Java específicas para celulares e *pagers*, entre elas destacam-se:

- Manipulação da entrada de dados;
- Manipulação da tela do dispositivo provendo alguns métodos para desenho;
- Especificação mínima de memória, resolução de vídeo, etc;

- Definição do formato em que as aplicações devem ser desenvolvidas a fim de serem executadas pelos dispositivos;

Baseado nesse *profile* a Sun implementou um kit de desenvolvimento [16] que permite a criação e execução de aplicativos baseados em MIDP.

A figura abaixo mostra o ambiente de desenvolvimento, onde pode-se observar que um mesmo programa é executado por mais de um dispositivo. Isso é possível porque o kit é capaz de emular diversos dispositivos que suportam MIDP (de um celular simples com tela monocromática a um Palm).



2.2 MIDP x Desenvolvimento de Jogos

Antes de explicar a especificação do wGEM, é conveniente discutir sobre os recursos presentes no MIDP com os quais se pode contar para especificação.

Do ponto de vista de desenvolvimento de jogos, os recursos do MIDP são poucos. Além do básico, que é prover acesso à tela do dispositivo para desenho e acesso ao teclado para

capturar eventos, restam poucos recursos que se comparados aos encontrados em J2SE, por exemplo, em que existem recursos para manipulação de polígonos, interseção de retas, suporte a som, etc.

A seguir são listados alguns recursos não presentes que resultaram em um maior esforço para realização de certas tarefas e que tornaram impossíveis a realização de outras:

- Ausência de ponto flutuante. O que obriga o programador a adaptar algoritmos já conhecidos para possibilitar o trabalho com aproximações inteiras;
- Acesso à cor de um pixel da tela. O que dificulta bastante a execução de algoritmos que dependem do conteúdo da tela;
- Ausência de som. Tornando o jogo mais monótono;
- Baixo poder de processamento. Inviabilizando a implementação de jogos mais complexos;
- Pouca memória. Ruim para jogos cujo estado é complexo e necessita armazenar muitos dados;

3 wGEM

O *framework* wGEM é resultado de uma generalização e adaptação de características encontradas em *frameworks* de desenvolvimento de jogos para PCs às limitações dos dispositivos móveis e da tecnologia J2ME. Consistindo de uma API para desenvolvimentos de jogos para dispositivos móveis utilizando J2ME/MIDP. Na tabela abaixo são descritos os módulos que ele implementa.

Objeto do Jogo	Representação do objeto que contém as informações necessárias para o seu gerenciamento (como posição e representação gráfica) e também as funcionalidades necessárias para a execução do jogo (como detectação de colisão).
Mapa do Jogo	*O framework* provê a utilização de mapas para facilitar o desenvolvimento de jogos com cenários complexos. Recursos como *scrolling* e mapas baseados em *tiles são* oferecidos
Gerenciamento de Objetos	É responsável por inicializar, armazenar e controlar o acesso aos objetos do jogo que ele gerencia.
Gerenciamento de Entrada	Identifica eventos do teclado do dispositivo e encaminha o ocorrido para o gerenciador do jogo.
Gerenciamento Gráfico	Realiza a apresentação gráfica do ciclo do jogo, coordenando o processo de desenho do mapa e dos objetos do jogo.
Gerenciamento do Jogo	Implementa o ciclo do jogo, utilizando os módulos de gerenciamento de objetos, do mapa do jogo, de entrada e gráfico.
Gerenciamento de Rede	Implementa as funcionalidades de *download* e *upload* de arquivos utilizando o protocolo HTTP suportado pelo perfil MIDP.
Gerenciamento de Aplicação	Implementa as funcionalidades exigidas pela KVM às aplicações do perfil MIDP para que estas possam ser gerenciadas.
Integração com Editor de Cenários	Oferece as funcionalidades necessárias para importação de mapas produzidos por meio da utilização do editor de cenários.

Vale ressaltar que devido ao fato de que alguns dos requisitos dos módulos de um *framework* serem bastante independentes, um *framework* de jogos também pode ser visto como a composição de diversos *frameworks* com papéis bem definidos. Em conseqüência dessa grande variedade de requisitos dos *frameworks*, muitos procuram limitar-se a determinados estilos de jogos. O wGEM, por exemplo, envolve apenas o desenvolvimento de jogos 2D para dispositivos móveis.

Já foram implementados vários jogos de gêneros diferentes utilizando o *framework* wGEM para exemplificar sua utilização e validar sua implementação, como pode ser visto nas figuras abaixo.

Jogo Breakout	Jogo Ship	Jogo Enduro

3.1 Pendências do wGEM

Embora o wGEM ofereça um bom framework para o desenvolvimento de jogos para dispositivos móveis, ele não é completo, não oferecendo alguns módulos necessários para a criação de jogos mais complexos.

Dentre as principais deficiências podemos destacar a falta de um módulo que implemente funcionalidades relacionadas a Inteligência Artificial.

Um outro ponto a ser melhorado no wGEM de modo a permitir uma maior complexidade dos jogos sem gerar uma carga tão grande no desenvolvedor é a sua modelagem física. A implementação dos objetos no wGEM leva em consideração apenas as velocidades do objeto nos eixos x e y; embora isso seja o suficiente para alguns jogos simples, jogos mais complexos precisam de um representação melhor.

Outros problemas que devem ser endereçados no futuro são relacionados a performance, pois embora várias otimizações já tenham sido realizadas, alguns pontos ainda podem ser melhorados. E também com relação ao tratamento de erros.

4 Extensão proposta ao wGEM

Nessa seção serão descritas as propostas de extensão ao *framework* wGEM.

4.1 Objetivos

Nesse trabalho propomos algumas extensões ao *framework* wGEM com o intuito de melhorar a qualidade/jogabilidade dos jogos desenvolvidos utilizando este sendo elas uma melhor modelagem física e a implementação de comportamentos de movimento.

4.2 Motivação

Seguindo a sugestão de trabalhos futuros no wGEM, partimos para a implementação de comportamentos inteligentes, e para alcançar esse objetivo precisamos também melhorar a modelagem física do *framework*. Detalharemos a seguir as motivações para a escolha destas extensões.

4.2.1 Modelagem Física

A principal motivação para a modelagem física é tornar a movimentação dos objetos do jogo mais verossímel e mais importante ainda acreditável. Entre os conceitos modelados se encontram massa, forças, velocidade e aceleração.

Além disso essa modelagem pode ser usada como base para desenvolvimento de camadas mais complexas, como por exemplo na implementação de comportamentos.

4.2.2 Comportamentos Inteligentes

Um ponto importante em jogos é a maneira com que os objetos/personagens se comportam e em particular como eles se movimentam na tela. Podemos dividir este gerenciamento de movimento em três níveis: estratégia, comportamento e locomoção

Sendo a locomoção a camada de baixo que representa o corpo do objeto, convertendo sinais do modelo de comportamento em movimentação, sendo essa sujeita a restrições do modelo físico do corpo (por exemplo, velocidade máxima) e é implementado através da modelagem física aqui apresentada.

O comportamento é exatamente a área onde vamos nos focar nesse trabalho. Alguns exemplos desses comportamentos são: perseguir ou fugir de um determinado objeto, seguir o líder, evitar colisões, andar em grupo, seguir um fluxo, etc.

A estratégia é a camada mais alta onde se faz o planejamento do que se vai fazer, que objetivos se quer atingir; ela não será coberta diretamente neste trabalho, mas combinações de comportamentos podem servir para implementar alguma estratégia ou alcançar um objetivo.

Com relação comportamentos, grande parte das questões sobre movimento de unidades hoje em dia gira em torno de minimizar a carga da CPU *versus* maximizar a exatidão e inteligência do movimento. Outra preocupação é a de implementar comportamentos inteligentes bem definidos e de uma maneira que possam ser reutilizados de modo que possam ser úteis para melhorar a experiência do jogador [10].

As pesquisas sobre esse assunto são relacionadas com robótica, IA e vida artificial, sendo geralmente voltadas para animação, ou seja, modelagem de comportamento está localizada na interseção entre animação e os três campos citados. O que de certo modo se encaixa com nosso intuito de trabalhar na expansão de um *framework* para desenvolvimento de jogos.

O método mais utilizado para implementar essas características é chamado *Steering behaviors*[2] e consiste em usar regras simples que juntas dão a grupos de agentes autonomos uma forma mais realística de movimentação por unidade e de comportamento em grupo. É uma técnica que não preserva estado, economiza memória, reage em tempo real, o que encaixa nos nossos requisitos já que temos limitação de memória e processamento em dispositivos móveis. Além do que (como dito anteriormente), combinações de *steering behaviors* podem ser usados para se alcançar objetivos de mais alto nível (por exemplo, atravessar uma região evitando obstáculos)

4.3 Estado da arte

4.3.1 Modelagem física

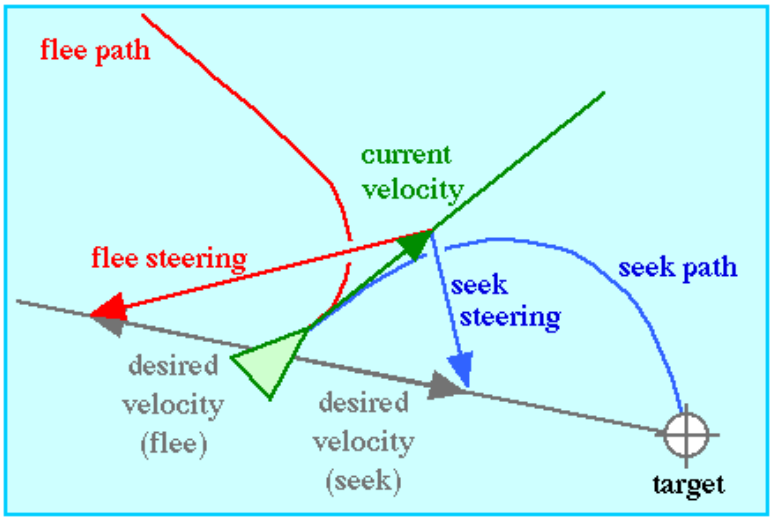
Com relação a modelagem física, não há muita informação disponível sobre como implementá-la, a literatura é bastante superficial se referindo apenas a comentar sobre a sua importância e os motores existentes que oferecem modelagem física ou não são utilizáveis ou são complexos demais para nossos propósitos. Para o tipo de ambiente no qual nos focamos, não há nada na literatura e os modelos propostos foram criados com base em nossas reflexões e na literatura existente sobre modelagem física.

4.3.2 Steering Behaviors

Com relação a coordenação de movimento, praticamente todas as implementações são baseadas em [2]. Partiremos agora então para descrever alguns modelos de comportamento propostos por Reynolds.

4.3.2.1 Comportamentos individuais

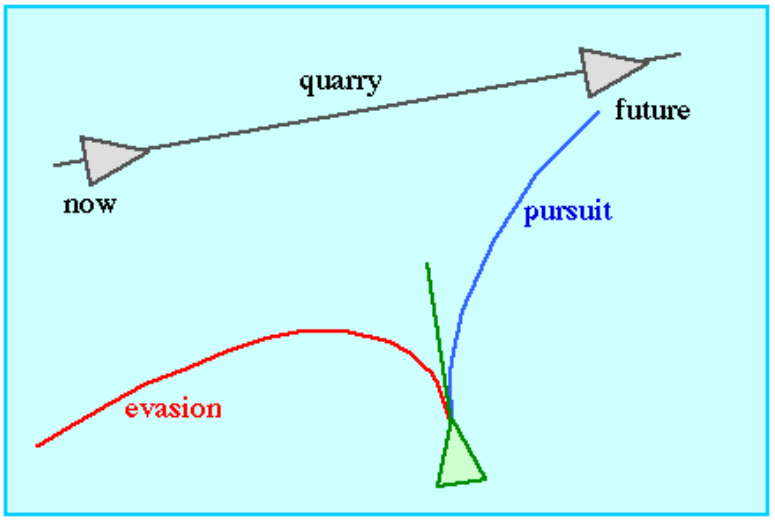
Seek e Flee



Seek funciona "puxando" o objeto em direção a um ponto específico no mapa. Esse comportamento ajusta a velocidade do objeto que ela fique radialmente alinhada em direção ao alvo. A "velociade desejada" é um vetor na direção do objeto para o alvo. E o módulo deste vetor pode ser expresso pela velocidade máxima, ou pela velocidade atual dependendo da aplicação. O vetor de steering é a diferença entre essa velocidade desejada e a velocidade atual do objeto.
Se o objeto sempre segue o comportamento **seek**, ele eventualmente vai passar pelo alvo, dar a volta e se aproximar novamente, produzindo um movimento semelhante ao de uma mosca.

Flee é simplesmente o inverso do comportamento anterior e puxa o objeto de modo que sua velocidade é radialmente alinhada para longe do alvo e a velocidade desejada é na direção oposta.

Pursuit e Evasion

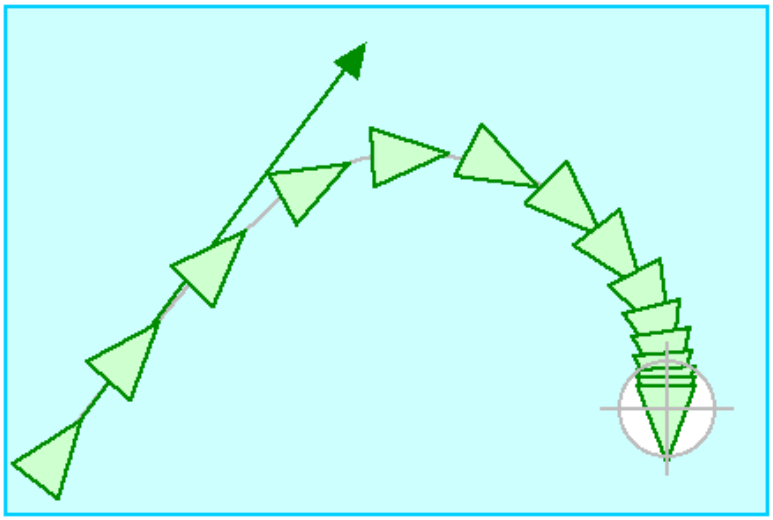


Pursuit é similar ao **seek** exceto pelo fato de que o alvo agora é um objeto móvel. Uma perseguição efetiva requer predição da posição futura do alvo. A idéia aqui é simplesmente prever a nova posição do avlo e reavaliar a cada passo. A predição corresponde a assumir que o alvo não vai mudar de direção durante o intervalo de predição. Isso quase sempre não é o que acontece, mas como esse resultado só será utilizado num curto espaço de tempo isso não é tão prejudicial e nos poupa de problemas relacionados a performance que surgiriam no caso de se tentar utilizar predições mais elaboradas. **Pursuit** é então simplesmente aplicar **seek** ao alvo previsto.

A chave do **pursuit** é o método usado para estimar o intervalo de predição. Idealmente, esse intervalo deveria ser o tempo até a interceptação, mas esse valor não é calculável pois o alvo pode fazer manobras arbitrarias e imprevisíveis. Uma abordagem seria esse tempo ser tratado como constante, pois embora ingênuo ainda seria melhor que um simples **seek**. Para uma perfomance razoável o tempo deveria ser proporcional a distância do alvo, grande quando longe do alvo e pequeno quando perto.

O comportamento **Evasion** é análogo a **pursuit**, exceto que **flee** é utilizado ao invés de **seek** para calcular a nova direção do objeto.

Arrival

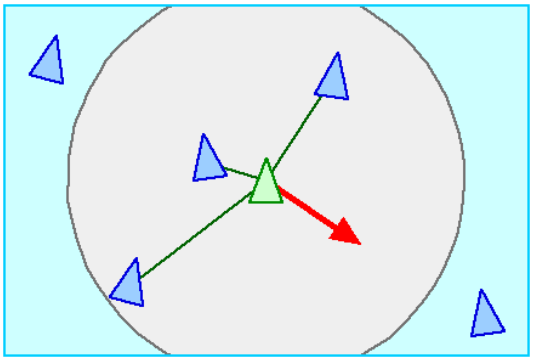


Arrival funciona identicamente a **seek** enquanto o objeto está longe do alvo. Mas ao invés de ter o problema de passar direto pelo alvo, o objeto vai freando a medida que se aproxima do seu alvo, eventualmente parando na posição desejada. A distância onde se deve começar a frear (ou raio de ação) é um paramêtro do comportamento.

4.3.2.2 Comportamentos de grupo

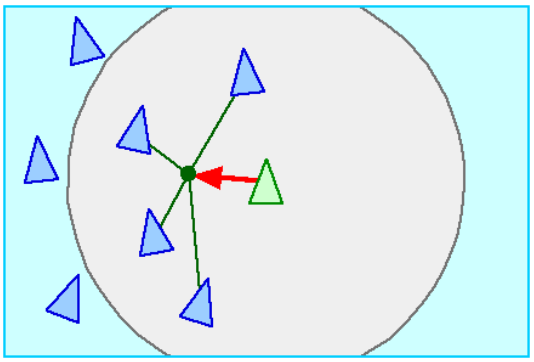
Além dos comportamentos individuais, há também comportamentos relacionados a grupos de objetos como, por exemplo, os próximos 3 comportamentos[1]: separation, cohesion e alignment.

Separation



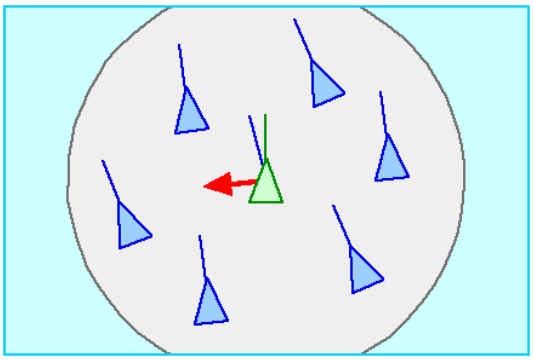
Oferece ao objeto a habilidade de manter uma certa separação dos outros próximos. Isso pode ser usado para impedir que objetos se aglomerem. Para computar a força do comportamento, é feita uma busca pelos objetos vizinhos. Para cada objeto próximo, uma força repulsiva é calculada subtraindo-se a posição do objeto da do seu vizinho, normalizando, e então se aplicando um peso. Essas forças repulsivas são somadas e a resultante é a força do comportamento.

Cohesion



Permite ao objeto se aproximar e formar um grupo com outros objetos próximos. Pode ser computada, encontrando os vizinhos de maneira similar a anterior e calculando a posição média (ou “centro de gravidade”) destes. A força de coesão será então aplicada na direção dessa posição, ou esta pode ser usada como alvo para **seek**.

Alignment



Provê a habilidade de um objeto se alinhar (apontar para a mesma direção) com outros objetos próximos. E pode ser calculado encontrando-se os objetos na vizinhança e tirando a média de suas velociades. Essa média é a velocidade desejada, então a força será calculada como a diferença entre a velocidade atual e esta velocidade desejada. Essa força tende a “virar” o objeto para a mesma direção para a qual os demais estão voltados.

4.3.2.3 Combinação de comportamentos

Os comportamentos descritos acima além de modelarem um comportamento específico, servem como blocos para definir novos padrões de comportamento mais complexos. Para se conseguir comportamentos interessantes e críveis, temos que selecionar e combinar estes components individuais, uma vez que a não ser que o “objeto” exista num mundo extremamente simples, não faz sentido ele apresentar sempre o mesmo comportamento.

A combinação de comportamentos pode ser conseguida de várias maneiras. A mais simples é calcular as forças dos comportamentos individuais e somá-los, possivelmente com algum fator de peso. Essa combinação linear simples funciona razoavelmente bem, mas tem dois "problemas": não é computacionalmente eficiente e apesar do ajuste dos pesos, as forças dos comportamentos podem se anular em momentos inoportunos.

4.3.2.4 Dificuldades

As limitações de J2ME e das plataformas de dispositivos móveis impõem certas dificulades para a implementação deste tipo de modelo, desde ausência de ponto flutuante e bibliotecas de funções matemáticas não disponíveis, a escassez de memória e tempo de CPU. Precisamos tomar muito cuidado para não gerar um *overhead* muito grande no sistema.

5 Implementação e Resultados

A implementação desse trabalho consistiu de extensões ao framewok wGEM (em especial por meio modificações nas representações dos objetos e nos gerenciadores de objetos) de modo a prover as funcionalidades necessárias à modelagem física e a implemetação de comportamentos inteligentes de movimentação.

5.1 Modelagem física

Um objeto wGEM antes, só tinha informação sobre sua posição e velocidade, sendo necessário então se especificar estruturas para representar outras características importantes e também as forças agindo sobre o objeto. Deste modo, um objeto do jogo passa agora a ter também massa, aceleração e uma lista de forças que agem sobre ele.
Foi também definida uma classe Vector2D que representa um vetor em duas dimensões, assim como implementa operações com vetores (tais como: soma, escala, normalização, etc.). Esta classe é utilizada para representar as forças exercendo influência no objeto.
O número de forças agindo sobre um único objeto foi limitada a 4 pois embora a modelagem física seja importante para melhorar a verossimilhança do jogo, não é necessário que este se comporte de maneira estritamente fiel ao mundo real, sendo inclusive (em alguns momentos) melhor para a jogabilidade que os objetos tenham comportamentos não possíveis na realidade (mas minimamente críveis). Outros motivos para essa simplificação são a queda de performance quando se tem vários objetos na tela e o fato de que para a maioria das situações presentes em um jogo, diversas forças podem ser desprezadas.
Com o intuito de “facilitar a vida“ dos desenvolvedores foram criadas classes que implementam algumas forças básicas, como atrito e gravidade.

5.2 Comportamentos de movimentação

Usando estes aspectos da modelagem física como a camada de base passou-se a trabalhar no nível acima, o dos comportamentos (steering behaviors). Cada objeto do jogo passa a poder aresentar um comportamento.

A cada passo da simulação, as forças determinadas pelo comportamento são aplicadas ao objeto, produzindo uma aceleração igual a força dividida pela massa do objeto. Essa aceleração é então somada a antiga velocidade para produzir uma nova (sendo essa truncada pela velocidade máxima) e finalmente adicionada à posição antiga.

Desse modo os steering behaviors são implementados através cálculos com vetores geométricos representado a força de steering desejada.

No modelo simples aqui implementado, o sinal de controle passado pelo comportamento é um vetor força. Modelos mais realísticos poderiam utilizar vários sinais, podendo mapear uma destas forças diferentemente dependendo do sinal (por exemplo, acelerando ou desacelerando) e não necessariamente simétircos (novamente utilizando o exemplo do carro, um carro desacelera mais rápido que acelera).

Alguns comportamentos também foram implementados (seek, flee, arrival, pursuit, evasion, cohesion, separation e alignment).

5.3 Estudos de caso

Como estudos de caso foram desenvolvidos alguns demos para validar a implementação e demonstrar a funcionalidade dos comportamentos implementados.
O desempenho dos modelos foi bem satisfatório quando o número de objetos utilizados não é muito grande.

Um protótipo de jogo utilizando estes comportamentos também está em desenvolvimento.

6 Conclusões

Este trabalho implementou alguns aspectos de modelagem física de modo a propiciar uma base para que desenvolvedores possam utilizar essas caracteristicas em seus jogos.
Além disso, foram implementados também comportamentos de movimentação que tiram vantagem dessa modelagem física, facilitando a implementação de objetos com comportamentos mais elaborados e verossímeis melhorando a experiência dos jogos.

Dificuldades

Uma das principais dificuldades encontradas nesta pesquisa foi certamente a ausência de literatura de qualidade sobre o funcionamento de modelos físicos em jogos, a maior parte da literatura trata o assunto muito superficialmente. Este fator aliado a necessidade de prover uma solução geral, levou a uma considerável necessidade de tempo para projeto.

As limitações técnicas da tecnologia J2ME consituíram também uma grande dificuldade, desde a familiarização com alguns dos conceitos à necessidade de implementação de algumas funcionalidades que normalmente já seriam oferecidas pela linguagem de programação.

Trabalhos Futuros

Como continuação do trabalho sobre o wGEM, diversas evoluções podem ser consideradas, divididas principalmente em extensões e otimizações.

No que diz respeito às extensões da arquitetura, as principais possibilidades são:

- Permitir objetos cuja representação gráfica seja baseada em polígonos;
- Melhorar o tratamento de exceções para facilitar o entendimento/localização de problemas ocorridos em seu funcionamento;
- Criação de um módulo voltado para inteligência artificial;
- Criação de um gerenciador de múltiplos jogadores;
- Definição de novos comportamentos e de maneiras de combinar diversos comportamentos

Em se tratando de *otimizações* do wGEM, acreditamos que o trabalho deva ser focado no gerenciador de objetos, o modelo utilizado por este gerenciador produz um sério *overhead* quando é preciso correr(a cada ciclo do jogo) os os objetos do repositório para descobrir alguma informação sobre os mesmos (por

exemplo, quais estão visíveis, quais colidem com um dado objeto, quais estão proximos de um certo objeto) uma vez que é necessário o acesso a todos os objetos do repositório. Também merecem atenção com relação a performance a implementação dos comportamentos inteligentes.

Uma outra abordagem seria a transcrição do *framework* para J2SE de modo que possa ser executado em dispositivos mais potentes.

7 Bibliografia

[1] Reynolds, C. W. (1987), `*Flocks, Herds, and Schools: A Distributed Behavioral Model*', Computer Graphics 21(4), 25--34.

[2] Reynolds, C. (1999) "Steering Behaviors For Autonomous Characters", URL: http://www.red3d.com/cwr/steer/gdc99/index.html Consultado em 20 de abril de 2002

[3] Lamothe, A. Tricks of the windows game programming gurus. SAMS Publishing. 1999

[4] BREW Home, URL: http://www.qualcomm.com/brew/ . Consultado em 26 de abril de 2002.

[5] TTPCom's Wireless Game Engine, URL: http://www.ttpcom.com/ttpcom/games.htm . Consultado em 26 de abril de 2002.

[6] DeLoura, M. Game Programming Gems. Charles River Media. 2000

[7] Mobile Information Device Profile (MIDP), URL: http://java.sun.com/products/midp/

[8] Wireless Developer, URL: http://wireless.java.sun.com/ . Consultado em 26 de abril de 2002.

[9] GDC 2001 conference proceedings. Game Developer Conference. 2001

[10] Pottinger D. "Coordinated Unit Movement", URL: http://www.gamasutra.com/features/19990122/movement_01.htm . Consultado em 20 de abril de 2002.

[11] Java™ 2 Platform, Micro Edition (J2ME™), URL: http://java.sun.com/j2me/

[12] JAVA™ 2 PLATFORM, STANDARD EDITION, URL: http://java.sun.com/j2se/

[13] JAVA™ 2 PLATFORM, ENTERPRISE EDITION, URL: http://java.sun.com/j2ee/

[14] Wireless games may be catching hold, URL: http://news.com.com/2100-1033-257622.html?legacy=cnet . Consultado em 24 de abril de 2001.

[15] Java™ 2 Platform Micro Edition (J2ME™) Technology for Creating Mobile Devices, URL: http://java.sun.com/products/cldc/wp/KVMwp.pdf

[16] Java™ 2 Platform Micro Edition, Wireless Toolkit, URL: http://java.sun.com/products/j2mewtoolkit/

[17] Rollings, A., Morris, D. Game Architecture and Design. Coriolis Group Books. 1999

[18] Briggs, J., *J2ME: The next major games platform?*, URL: http://www.javaworld.com/javaworld/jw-03-2001/jw-0309-games_p.html. Consultado em 24 de abril de 2001.

[19] Wireless gaming worth US$6 billion by 2005, URL: http://www.datamonitor.com/press/prtemplate.asp?id=tc000905+Wireless+Gaming

[20] Pessoa, C. wGEM: um Framework de Desenvolvimento de Jogos para Dispositivos Móveis – Dissertação de Mestrado

[21] J2ME Connected Device Configuration, URL: http://java.sun.com/aboutJava/communityprocess/first/jsr036/index.html

[22] J2ME Connected, Limited Device Configuration, URL: http://java.sun.com/aboutJava/communityprocess/final/jsr030/index.html