Análise de tecnologias de desenvolvimento de jogos para dispositivos móveis

BÖRJE FELIPE FERNANDES KARLSSON
IVO FRAZÃO NASCIMENTO
TIAGO GUEDES FERREIRA BARROS
GEBER LISBOA RAMALHO

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – CEP 50.732-970 – Recife – PE – Brasil
{bffk, ifn, tgfb, glr}@cin.ufpe.br

Abstract

O presente artigo faz uma análise das "plataformas" de desenvolvimento de jogos para dispositivos móveis mais difundidas atualmente. Apresentando comparações entre elas com relação a arquitetura de desenvolvimento, funcionalidades disponibilizadas e facilidade de desenvolvimento. Além de analisar em que situações cada uma delas melhor se adequam.

Keywords: *Desenvolvimento de jogos, Celulares, Jogos em dispositivos móveis.*

1 Introdução

O mercado de jogos é uma área em franca expansão, movimentando anualmente milhões de dólares no mundo todo[19]. O mercado de dispositivos móveis, por sua vez, também se econtra em franca expansão, com os aparelhos celulares deixando de ser simplesmente aparelhos telefônicos para tornarem-se agendas, leitores de e-mail, tocadores de MP3, equipamentos com uma faceta multifuncional.

Não é de se surpreender que também sejam explorados pela indústria de entretenimento para a distribuição de jogos. São jogos que podem ser jogados em qualquer lugar a qualquer momento. Estimativas [18] apontam para um faturamento do mercado de jogos para dispositivos móveis entre 3,5 e 4,4 bilhões de dólares em 2006, chegando a 9 bilhões de dólares em 2008.

Inicialmente, as próprias empresas que fabricavam os aparelhos também desenvolviam todos seus aplicativos. Mas com a grande demanda para o desenvolvimento de novas e, cada vez, mais complexas aplicações (principalmente jogos), percebeu-se a necessidade da criação de um meio que permitisse que outras empresas pudessem desenvolvessem estas aplicações.

Por causa desta necessidade, diversos fabricantes e empresas de software decidiram construir camadas de software entre os sistemas operacionais de seus aparelhos e as aplicações, permitindo que outras empresas desenvolvessem sobre utilizando esta camada para acessar as funcionalidades nativas dos aparelhos além de prover uma base consistente entre diversos aparelhos, de modo a facilitar o desenvolvimento para varias plataformas.

A primeira iniciativa neste sentido foi Java 2 Micro Edition (J2ME), uma versão da linguagem Java para dispositivos móveis. Desta forma, os fabricantes poderiam implementar as máquinas virtuais onde o código Java seria executado, uma abordagem já utilizada no uso de aplicações Java em navegadores (os Applets Java). A adoção desta solução permitiu o desenvolvimento de um número muito maior de aplicações em menos tempo e com a grande disponibilidade de programadores que conheciam a linguagem Java (versão Standard), que poderiam migrar para facilmente para J2ME, possibilitou a diminuição do custo de desenvolvimento das aplicações.

Seguindo esta filosofia, a Qualcomm lançou BREW (Binary Runtime Enviroment for Wireless), um ambiente para execução de aplicativos pré-compilados, para seus microcontroladores CDMA, e disponível na maior parte dos aparelhos que utilizam esta tecnologia. Outras iniciativas também foram lançadas (como por exemplo, [14][15][16]) mas estas não alcançaram grande repercurssão.

Paralelamente a esta situação, foi criado o Symbian, um consórcio formado pelas maiores empresas da indústria de comunicação sem fio (Nokia, Panasonic, Motorola, Psion, Samsung, Siemens, Sony e Ericsson), para o desenvolvimento de um sistema operacional comum a seus dispositivos, o SymbianOS. Este sistema provê também um framework de desenvolvimento de aplicações que dá suporte ao desenvolvimento de jogos[2]. Além de ser o sistema operacional do primeiro dispositivo móvel do tipo telefone celular dedicado a jogos, o Nokia N-Gage[17].

Desenvolver um jogo não é uma tarefa trivial, pois envolve o conhecimento profundo de diversas áreas da computação e da tecnologia usada no seu desenvolvimento. Em ambientes restritos, com limitações de memória e processamento, esta tarefa é ainda mais difícil. Vários frameworks para as várias tecnologias já foram desenvolvidos[1][5][6] e extendidos[7] para tentar diminuir essas dificuldades.

Com base nesse cenário partimos para um estudo comparativo das plataformas (embora não sejam exatamente plataformas, estamos chamado-as assim pois a comparação é aqui feita do ponto de vista do desenvolvedor de jogos) mais utilizadas. Descreveremos então J2ME, BREW e o SymbianOS. E finalmente concluímos com uma comparação das características de cada tecnologia e a que situações cada uma melhor se adapta, de modo a ajudar os desenvolvedores de jogos para sistemas móveis a definir qual plataforma melhor se adequa a seus projetos.

2 J2ME

Criada a partir da decisão da Sun de dividir a tradicional plataforma Java utilizada nos PCs em três partes, cada uma sendo direcionada para diferentes categorias de dispositivos, Java 2 Micro Edition (J2ME)[13], foi direcionada para o desenvolvimento de aplicações voltadas para dispositivos com capacidade limitada de processamento e memória. Com a observação dessa variedade de dispositivos, J2ME por sua vez adotou uma arquitetura modular e escalável, consistindo de um conjunto de blocos que se complementam para representar as particularidades dos grupos de dispositivos existentes.

Estes blocos seriam três camadas:

i) a KVM, a máquina virtual em si.

ii) Camada de configuração, que define os recursos Java que fazem parte de uma larga categoria de dispositivos que compartilham certas características importantes, tais como capacidade de memória, comunicação, consumo de energia e interface com o usuário. Em outras palavras, uma configuração define o "menor denominador comum" dos recursos da plataforma Java que serão suportados em todos os dispositivos pertencentes à larga categoria considerada pela configuração.

iii) Perfil, que define a API voltada para demandas específicas de um segmento de mercado, tais como, carros, máquinas de lavar, celulares e televisores. Um perfil é a camada mais próxima para os usuários e desenvolvedores de aplicações. Ela provê mecanismos como entrada de dados, interface com o usuário, persistência de dados e conectividade da forma ideal para o segmento de mercado abrangido pelo perfil.

Quando falamos de J2ME para telephones celulares, estamos nos referindo a J2ME usando a configuração CLDC (Connected Limited Device Configuration) e o perfil MIDP (Mobile Information Device Profile) rodando sobre a KVM (Kilobyte Virtual Machine) (KVM). Essensialmente, CLDC e MIDP definem um conjunto de funcionalidades disponíveis para uma família de dispositivos (Figura 1).

Figura 1: J2ME

2.1 Desenvolvimento de aplicações em J2ME

As aplicações desenvolvidas usando MIDP recebem o nome de MIDlets. Basicamente o desenvolvedor só precisa herdar da classe MIDlet e implementar os métodos pauseApp, startApp e destroyApp. J2ME é uma das plataformas mais fáceis para se começar a desenvolver.

Um ponto importante a observar, é que devido a falta de certas características úteis a jogos no MIDP 1.0, diversos fabricantes e operadoras criaram suas próprias extensões, sendo necessário que o desenvolvedor conheça estas características para melhor tirar proveito das capacidades dos telefones.

2.2 Suporte de J2ME a jogos

Após conhecermos um pouco sobre J2ME, iremos detalhar quais os recursos específicos presentes na API que podem ser usados para auxiliar no desenvolvimento de jogos, bem como algumas de suas extensões. [8] apresenta mais detalhes.

2.2.1 Gerenciamento de I/O

J2ME dá suporte à manipulação do teclado dos dispositivos, que é um requisito fundamental para um jogo, dada a necessidade de interatividade com o jogador; o MIDlet pode tratar eventos das teclas através dos métodos keyPressed, keyRealeased e keyRepeated. Sendo as teclas tratadas como teclas normais ou GameAction no caso de direcionais ou tiro por exemplo.

2.2.2 Processamento Gráfico

J2ME provê suporte à manipulação da tela gráfica, fornecido pela classe Canvas do perfil MIDP, possibilitando o desenho de imagens e formas geométricas na tela do dispositivo. Oferecendo funcionalidades de desenho de primitivas gráficas, clipping, double-buffering e suportando imagens no padrão PNG (com transparência na grande maioria dos dispositivos). Atenção especial deve ser dada as paletas, que muitas vezes são hard-coded nos telefones, o que pode causar problemas de exibição de cores.

2.2.3 Processamento de Som

Infelizmente MIDP 1.0 não permite que o desenvolvedor faça uso de suporte a som. O que pode ser atenuado pelas inúmeras extensões dos fabricantes e operadoras. Suporte a som também pode estar disponível se o telefone suportar uma versão mais nova de MIDP ou a JSR 135, conhecida como Mobile Media API.

2.2.4 Conectividade

J2ME sobre MIDP 1.0 provê suporte à conectividade com outros dispositivos, possibilitando que jogos em rede sejam desenvolvidos através do protocolo HTTP fornecido pelo perfil MIDP.

Extensões como a JSR 120, conhecida como Wireless Messaging API aumentam essa capacidade acrescentando recursos de envio e recebimento de mensagens SMS.

2.2.5 MIDP 2.0

A nova versão da API de J2ME traz novos recursos para os desenvolvedores de jogos. Ela possui gerenciamento para Sprites, Mapas de Tiles, manipulação direta do RGB das imagens e até classes para checagem de colisão. Na área de conectividade, passa a dar suporte a sockets, datagramas UDP, HTTPS e server sockets aumentando muito o potencial para jogos multi-usuário.

Adicionalmente, MIDP2.0 também traz um subconjunto da JSR135 (Mobile Media API) garantindo suporte a MIDI e arquivos WAV. A JSR 135 por sua vez traz suporte a geração de tons. Suporta também playback e gravação de sons (mas os formatos dependem dos dispositivos), playback de vídeo e garante a execução de no mínimo quatro sons simultâneos. Está em fase de conclusão também a JSR 184 que trata de imagens 3D em dispositivos móveis.

2.2.6 Observações

Como J2ME é um padrão aberto, cada operadora de telefonia cellular ou fabricante de dispositivos pode crier suas próprias extensões. Além disso, cada operadora tem seus esquemas de teste e certificação o que gera a necessidade de se crier laços comercias e se testar o aplicativo para cada operadora.

3 BREW

BREW (Binary Runtime Enviroment for Wireless)[11], um ambiente para execução de aplicativos pré-compilados, foi lançado pela Qualcomm para permitir que desenvolvedores independentes produzissem aplicações para seus microcontroladores CDMA, largamente difundidos nos aparelhos celulares que adotam esta tecnologia.

A API que faz parte do ambiente BREW foi criado a partir de uma já utilizada internamente pelos desenvolvedores da Qualcomm para desenvolver suas próprias aplicações. Esta API foi aprimorada para permitir que outros desenvolvedores pudessem utiliza-la para criar suas próprias aplicações que usufruíssem dos recursos disponíveis nos aparelhos.

3.1 Desenvolvimento de aplicações em BREW

As aplicações desenvolvidas para BREW são escritas em C/C++, utilizando a API de BREW para acessar as funcionalidades dos aparelhos celulares. A grande maioria dos aparelhos celulares existentes no mercado dão suporte a versão 1.1 da API.

Para o desenvolvimento de aplicações em BREW é necessário que cada aplicação possua um identificador único e associado a este identificador existe ainda um arquivo descritivo da aplicação (arquivo .MIF) que descreve o nível de acesso aos recursos que determinada aplicação deve possuir. Este arquivo descreve ainda outras informações sobre a aplicação como nome, ícone, outras classes/aplicações/extensões utilizadas em conjunto.

Sua aplicação deve ser implementada como uma classe que herda de AEEApplet e deve implementar 3 métodos que serão responsáveis pela inicialização dos atributos, tratamento dos eventos e finalização da aplicação. Quando herdamos de AEEApplet, passamos a ter acesso as estrutura utilizadas pelas funções da API: m_pShell, usada pelas rotinas básicas da API e para ter acesso a outras interfaces; m_pDisplay, usada para ter acesso as rotinas de acesso a tela do dispositivo.

A execução das aplicações em BREW é baseada em eventos. Para cada ação executada pelo usuário ou pelo aparelho é informada através de um evento gerado que deve ser tratado pelo método da classe definido.

Após a execução das rotinas de instanciação, um evento de EVT_APP_START é gerado para indicar o início da aplicação. Ao fim da aplicação, um evento de EVT_APP_STOP é gerado. Além destes dois eventos, existem ainda os eventos de EVT_APP_SUSPEND e EVT_APP_RESUME, que são chamados quando alguma ação no aparelho celular deseja interromper a aplicação (o recebimento de uma chamada telefônica ou mensagem SMS). O pressionamento de teclas (EVT_KEY_PRESS e EVT_KEY_RELEASE) e a apresentação de dialogs (EVT_DIALOG_INIT, EVT_DIALOG_START e EVT_DIALOG_END) também geram eventos de início e fim.

Para aplicações que necessitam de atualização numa determinada freqüência, como jogos, a solução é utilizar um temporizador para invocar um método da classe após determinado intervalo de tempo.

3.2 Suporte de BREW a jogos

Após conhecermos o básico sobre aplicações em BREW, iremos detalhar quais os recursos específicos presentes na API de BREW que podem ser usados para auxiliar no desenvolvimento de jogos.

3.2.1 Gerenciamento de I/O

BREW trata o pressionamento das teclas como mais um evento para ser tratado pela sua aplicação. Como outros ambientes de execução para dispositivos móveis, BREW não dá suporte ao pressionamento de múltiplas teclas. Existem definidos 3 eventos para o tratamento de teclas:

EVT_KEY_PRESS – gerado quando uma tecla é pressionada, o método de tratamento de eventos recebe também um inteiro representando a tecla pressionada;

EVT_KEY_RELEASE – idêntico ao evento EVT_KEY_PRESS, mas é gerado quando a tecla deixa de ser pressionada;

EVT_KEY – idêntico a EVT_KEY_PRESS, deve ser usado quando o evento de liberação da tecla não precisa ser tratado;

3.2.2 Processamento Gráfico

BREW possui rotinas para a manipulação de imagens nos formato BMP e PNG, permitindo o clipping, blitting e a utilização de transparências e máscaras. Também existe um suporte para primitivas gráficas 2D, possibilitando o desenho de pontos, retas, círculos e polígonos arbitrários. Também dá suporte a transformações de visualização.

BREW ainda possui duas alternativas para trabalhar com animações:

utilizando o formato proprietário BCI da Qualcomm, que pode ser gerado a partir de um utilitário que vêm junto com o SDK de desenvolvimento;

utilizando a interface IImage, que é a mesma usada para carregar as imagens não-animadas. Basta carregar uma imagem e determinar quantos frames ela possui, depois disso usasse as funções IIMAGE_Start() e IIMAGE_Stop(), para, respectivamente, iniciar e parar a animação. Existe ainda a possibilidade de usar a função IIMAGE_DrawFrame(), para desenhar um frame em específico;

Mas não é possível acessar os pixels da tela ou das imagens, assim como algumas operações como flipping ou opção de double buffering não estão disponíveis.

3.2.3 Processamento de Som

BREW permite que sejam tocados beeps, rings e diversos tons. Existe a possibilidade de definir diversos tons para serem executados ao mesmo tempo. Existe ainda controle sobre volume e suporte a execução de arquivos MIDI, MP3 e QCP (Qualcomm PureVoice).

3.2.4 Conectividade

BREW possui suporte a criação de sockets utilizando TCP ou UDP. Permitindo assim a criação de aplicações cliente-servidor ou peer-to-peer. Existe ainda uma interface completa para transações web, com suporte a requisições assíncronas, serviço de proxy, conexões keep-alive entre outras coisas.

3.3 BREW 2.0

A nova versão da API de BREW traz novos recursos para os desenvolvedores de jogos. Ela possui gerenciamento para Sprites e para Mapa de Tiles com possibilidade tiles de diversos tamanhos. As transformações geométricas também foram melhoradas, permitindo agora escalas, rotações e flipping de imagens. Também é possível ter acesso aos pixels de uma imagem DIB (Device Independent Bitmap) além de acesso ao frame-buffer da imagem, o que pode dar início a utilização de efeitos mais efeitos mais elaborados.

BREW 2.0 também permite ao usuário instalar via aérea uma máquina virtual Java, permitindo assim a execução de MIDlets sobre BREW. No entanto são poucos os telefones no mercado com suporte a BREW 2.0.

3.4 Distribuição

Um ponto forte de BREW mas não tão ligado a tecnologia em si, é o seu modelo de negócios/distribuição, o BDS (BREW Distribution System), onde por ser uma tecnologia da Qualcomm essa tem completo controle sobre as aplicações. E estas, uma vez passando os testes de certificação da Qualcomm ficam disponíveis num repositório central para todas as operadoras podendo o desenvolvedor escolher facilmente o sistema de tarifação.

4 Symbian

Estabelecido como uma empresa independente em 1998, Symbian é um consórcio formado pelas maiores empresas da indústria de comunicação sem fio para o desenvolvimento de um sistema operacional comum a seus dispositivos, o SymbianOS. A intenção destas empresas foi desenvolver um sistema operacional para dispositivos móveis que seja aberto e padrão, a partir do sistema operacional EPOC, para PDA's. Devido a estas características, tornou-se possível o desenvolvimento de aplicações, por parte do usuário, para os dispositivos que possuíssem este sistema operacional. Então, agora os desenvolvedores podem criar aplicações em código nativo, para ser executado pelo processador do dispositivo.

4.1. Desenvolvimento de aplicações para SymbianOS

O sistema operacional Symbian permite o desenvolvimento de aplicações em C++, as quais são carregadas no dispositivo em forma de DLL. O SymbianOS pode ter vários tipos de interface gráfica com o usuário, dentre os quais, os mais comuns são:

UIQ[9]: interface para dispositivos que possuem uma tela com dimensões entre 4x6 cm a 6x8 cm e touch screen (tela sensível ao toque). O Sony Ericsson P800 (Figura 2) é um exemplo dispositivos com esta interface.

Series 60: de acordo com [3] e [4], a plataforma “Series 60” é um padrão de interface com o usuário criada pela Nokia, mas aberta a todos os dispositivos que utilizem o SymbianOS. É voltada para smartphones, que possuem uma tela de 176x208 Pixels. O N-Gage (também mostrado na Figura 2) é um exemplo.

Figura 2: O Sony Ericsson P800 (UIQ) e o Nokia N-Gage (Series 60).

A base do framework para desenvolvimento de aplicações para o SymbianOS é o UIKON – User Interface and Control Framework (framework de controle e interface com o usuário), que é comum às duas interfaces (UIQ e Series 60). Entretanto, para cada tipo de interface com o usuário é criada uma extensão do UIKON, a plataforma Series 60 utiliza o AVKON, enquanto o UIQ utiliza o QIKON. Podemos encontrar maiores detalhes sobre as diferenças entre estes frameworks específicos e como fazer o porting de um para o outro em [10].

Para utilizar o framework de aplicações Symbian, o desenvolvedor deve implementar as seguintes classes(Figura 3):

i) Classe da Aplicação, que provê a inicialização da aplicação, sendo a primeira classe a ser instanciada pelo framework. Esta classe instancia a classe de documento.

ii) Classe de Documento, utilizada para gerenciar a parte persistente (arquivos) da aplicação. É possível definir/associar quais os tipos de arquivos (extensões) são suportados/gerenciados por determinada aplicação. É importante notar que nem todas as aplicações possuem documentos associados, no entanto esta classe deve existir, pois esta classe instancia a classe de interface com o usuário.

iii) Classe de Interface com o Usuário (UI), provê ao desenvolvedor o tratamento de eventos como: pressionamento de teclas, posicionamento do ponteiro (no caso de touch screens), abertura/fechamento de arquivos, etc. Também é responsável pela criação da(s) classe(s) de visualização.

iv) Classe de Visualização (View), representa tudo o que está sendo mostrado na tela, recebendo da classe de UI os comandos para desenhar e atualizá-la. É importante salientar que uma aplicação pode ter várias views, sendo a classe de UI responsável por gerenciá-las e definir qual view estará ativa.

v) Motor ou núcleo da aplicação (ApplicationEngine), que implementada pelo desenvolvedor, é a que contém toda a lógica da aplicação, implementa seus algoritmos e provê os pontos de acesso para a classe de UI.

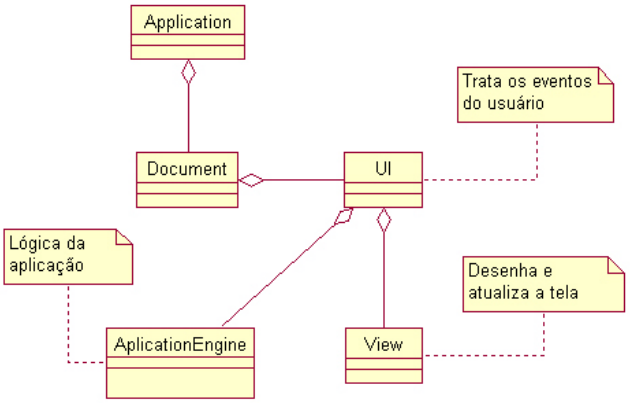

Figura 3: diagrama de classes do UIKON framework.

As classes de aplicação, documento, UI e view que devem ser desenvolvidas, herdam de classes correspondentes do UIKON (ou de uma de suas extensões: AVKON para Series 60 e QIKON para UIQ) e possuem

as suas funcionalidades básicas implementadas. Entretanto a classe núcleo da aplicação (ApplicationEngine) não possui correspondente no UIKON, devendo ser implementada pelo desenvolvedor.

4.2 Suporte de Symbian a jogos

Agora que sabemos como é o desenvolvimento de aplicações para o SymbianOS, iremos discutir quais as funcionalidades necessárias para o desenvolvimento de jogos que estão (e as que não estão) presentes na sua API.

4.2.1 Gerenciamento de I/O

O SymbianOS permite receber os eventos das teclas do telefone. São gerados eventos para pressionamento e liberação de teclas e, quando uma tecla permanece pressionada, um evento correspondente àquela tecla é enviado de acordo com a taxa de repetição de teclas do telefone. Apesar deste suporte, o SymbianOS não trata o pressionamento simultâneo das teclas. Portanto, isto limita um pouco a interação do usuário com o jogo, por exemplo: para mover um objeto na diagonal (por exemplo: para cima e para a esquerda ao mesmo tempo), é preciso definir uma tecla específica, já que não é possível tratar o pressionamento simultâneo das teclas "seta acima" e "seta esquerda".

4.2.2 Processamento Gráfico

A API do SymbianOS oferece diversas primitivas para desenho e processamento gráfico. Para acessar o display, a API disponibiliza um device context, classe que abstrai o dispositivo de saída (que pode ser a própria tela ou um bitmap na memória, por exemplo). Esta classe possui métodos para fazer desenho de primitivas gráficas (pontos, linhas, retângulos, etc), blitting (cópia de bits) com ou sem máscara, double buffering (desenhar em um buffer na memória e depois copiar este buffer para a tela) e outras operações com imagens. Entretanto, não suporta flipping (inverter um bitmap vertical ou horizontalmente), de forma que o desenvolvedor deve escrever o código para implementar isto ou criar as imagens invertidas. Os arquivos de imagens suportados são BMP, WBM (wireless bitmap) e JPEG.

4.2.3 Processamento de som

O SymbianOS permite tocar is seguintes formatos de arquivos de áudio: MIDI, WAV e AMR. Sua API disponibiliza classes que permitem o controle sobre a reprodução do áudio. Desta forma é possível alterar o volume, pitch, dentre outros atributos do som que se está tocando.

4.2.4 Conectividade

Por ser desenvolvido para celulares, o SymbianOS permite acesso à todas as formas de conexão destes dispositivos. Sua API fornece acesso a:

GPRS: é possível criar conexões via socket (TCP e UDP) e acessar a internet.

Bluetooth: podemos criar conexões pear-to-pear através de Bluetooth, o que para jogos é extremamente interessante, pois este tipo de conexão permite a criação de jogos multiplayer sem que o jogador tenha que pagar taxas de conexão (como no caso de GPRS). A desvantagem é que o alcance da conexão Bluetooth é pequeno (aproximadamente 10m).

SMS: o SymbianOS oferece uma API de acesso às funcionalidades de envio de mensagens através da rede de telefonia. Os primeiros jogos multiplayer criados para celulares usavam comunicação baseada na troca de mensagens, e esta API pode ser usada para o desenvolvimento de jogos neste estilo.

5 Conclusão

Como mostrado acima, cada tecnologia tem características importantes para desenvolvimento de jogos e proveem uma boa interseção do ponto de vista de funcionalidades. E todas as plataformas exigem testes constantes em telefones reais, não apenas nos emuladores.

J2ME oferece um ambiente orientado a objetos, com boa portabilidade entre dispositivos além de ser uma fácil opção de entrada por não estar atrelada a fabricantes de dispositivos. Além disso, existe uma grande base de desenvolvedores Java e J2ME é a tecnologia mais usada no mercado global (além de ser possível desenvolver usando J2ME sobre BREW e Symbian). J2ME também abstrai o usuário de problemas com alocação de memória e através das JSRs os dispositivos podem oferecer uma vasta gama de funcionalidades "padronizadas".

BREW oferece desenvolvimento em C/C++, sendo o que chega no mais "baixo-nível", mas não dá suporte total a orientação a objetos amarrando um pouco o desenvolvedor. Além de necessitar de ferramentas especiais

para compilar para a plataforam destino (processador ARM) e do pagamento de taxas de associação e testes que podem desmotivar muitos desenvolvedors. Vem crescendo bastante e BREW 2.0 parece ser uma plataforma bastante promissora. No entanto o grande atrativo de BREW é o sistema de distribuição e certificação centralizado, que conta com operadoras de peso como a Verizon (maior operadora celular dos EUA), Vivo (maior operadora celular da America do Sul) e KDDI (uma das maiores no Japão) facilitando a comercialização dos jogos no mundo todo.

SymbianOS é uma iniciativa dos principais fabricantes de dispositivos móveis, conta com telefones de última geração além de um dispositivo dedicado a jogos para dispositivos móveis (o N-Gage). Oferece também um framework de desenvolvimento que guia o desenvolvedor e evita que este possa comprometer o sistema. Sendo o ideal para a maioria dos jogos de maior porte.

6 Referências

[1] Pessoa, C. A. C., "wGEM: um Framework de Desenvolvimento de Jogos para Dispositivos Móveis". Dissertação de Mestrado, Centro de Informática - Universidade Federal de Pernambuco, Novembro de 2001.

[2] Planning a Game Application – Version 1.2, Nokia Corporation, Janeiro 2003. http://ncsp.forum.nokia.com/support/?body=detail&aid=86 (07/agosto/2003)

[3] Designing C++ Applications for Series 60 – Version 1.0. Nokia Corporation, Agosto 2002. http://ncsp.forum.nokia.com/downloads/nokia/documents/Designing_CPP_Applications_for_Series60.pdf (27/julho/2003)

[4] Series 60 Application Framework Handbook – Version 1.0. Nokia Corporation, Maio 2002. http://ncsp.forum.nokia.com/downloads/nokia/documents/Series60_Application_Framework_Handbook.pdf (27/julho/2003)

[5] Nascimento, I. F. "Desenvolvimento de um Framework para Jogos sobre a plataforma Brew", Trabalho de graduação, Agosto 2003.

[6] Barros, T. G. F. "SymbG(r)aF - Symbian Games Framework", Trabalho de graduação, Agosto 2003.

[7] Karlsson, B. F. F., Ramalho, G. L. "Incorporando Comportamentos de Movimentação e Modelagem Física ao wGEM", In Proceedings of WJogos 2002, Fortaleza, CE, Oct. 2002.

[8] Wabber Miranda de Arruda Filho and et. al. "Developing J2ME games: Problems and Saves", In Proceedings of WJogos 2002, Fortaleza, CE, Oct. 2002.

[9] UIQ - http://www.uiq.com/ (27/julho/2003)

[10] Porting Series 60 to UIQ. - http://www.symbian.com/developer/techlib/papers/series60-UIQ/Series60-UIQ.pdf (27/julho/2003)

[11] BREW., http://www.qualcomm.com/brew/ (05/agosto/2003)

[12] BREW and J2ME: A Complete Wireless Solution for Operators Committed to Java. White Paper. Qualcomm. Abril de 2003.

[13] Java 2 Platform, Micro Edition (J2ME), http://java.sun.com/j2me/ (05/agosto/2003)

[14] Developing Applications For The Microsoft Smartphone, http://msdn.microsoft.com/library/en-us/dnsmtphn/html/devappsp.asp (27/julho/2003)

[15] Mophun, http://www.mophun.com/main_developers.php (27/julho/2003)

[16] ExEn, http://developer.in-fusio.com/information_technology.php (27/julho/2003)

[17] Nokia N-Gage, http://www.n-gage.com (30/julho/2003)

[18] Miao, O., "Developing wireless games – from idea to the marketplace". In proceedings of the Game Developers Conference 2003

[19] Interactive Digital Software Association, "Essential facts about the computer and video game industry - 2003 Sales, Demographics and Usage data.", Maio 2003.