FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA – UNIFOR
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
CURSO CIÊNCIA DA COMPUTAÇÃO

Elaboração de um Estudo de Caso para Dispositivos Móveis Utilizando o Acelerômetro do Android na *Engine Unity*

Davi Mustafa de Oliveira

Fortaleza – Ceará 2013

Davi Mustafa de Oliveira

Elaboração de um Estudo de Caso para Dispositivos Móveis Utilizando o Acelerômetro do Android na *Engine Unity*

Monografia apresentada para obtenção dos créditos da disciplina Trabalho de Conclusão do Curso do Centro de Ciências Tecnológicas da Universidade de Fortaleza, como parte das exigências para graduação no Curso de Ciências da Computação.

Orientador:
Rafael Garcia Barbosa, M.Sc.

Fortaleza – Ceará 2013

Sumário

Introdução

No contexto de jogos digitais, as pessoas estão acostumadas com a execução de movimentos involuntários com um "*joystick*" na mão, com o qual o usuário costuma apertar botões para executar ações na tela do seu jogo. Apesar de esse cenário ter se tornado o mais natural, a evolução das tecnologias e a concorrência do mercado têm exigido novas formas de interação.

Atualmente com a criação de diversas formas de interação que vem sendo criadas, como a captura de movimento ou giroscópio, as imprecisões nos controles em alguns jogos acabam por tornar os mesmos frustrantes ou entediantes de serem jogados.

A tecnologia avançou a um ponto em que esse tipo de interação também pudesse ser vista e implementada em outros tipos de dispositivos. Por exemplo, empresas fabricantes de discos rígidos já estão fabricando os mesmos com sistemas de segurança próprios que detectam quando um *notebook* está em queda para proteger os dados do mesmo.

Atualmente, muitos *smartphones*, *vídeo games* e *tablets* usam pequenos sensores chamados acelerômetros, os quais permitem um maior controle na interface do usuário. Esses sensores basicamente reduzem a necessidade de botões de funções em dispositivos móveis. Muitos desenvolvedores usam o princípio do acelerômetro para a criação de aplicativos, como jogos, ou para controle da orientação da tela de exibição nesses dispositivos.

O objetivo desse trabalho é demonstrar a forma como o acelerômetro trabalha ao se desenvolver uma aplicação usando a plataforma Android em conjunto com a *engine Unity*.

Pesquisas têm sido realizadas nos últimos anos sobre aplicativos para dispositivos móveis, mas historicamente a falta de acesso ao *hardware* de um dispositivo tem sido frustrante para desenvolvedores que trabalham com essas plataformas. Ainda que as linguagens de desenvolvimento como Java e C# facilitem a comunicação do programador com o *hardware*, o time de desenvolvimento do Android trouxe muitos dos recursos do *hardware* para superfície.

Devido aos inúmeros avanços na área tecnológica, o desenvolvimento para dispositivos móveis tem apresentado um rápido crescimento e neste contexto o acelerômetro passou a ser um componente padrão nas aplicações Android. O interesse é justificado baseado na diversidade de dispositivos de sensoriamento de baixo custo disponíveis no mercado e avanços na área de robótica, como descrito no artigo por Figueireido [5].

Os *smartphones* mais usados atualmente são baseados em dispositivos eletromecânicos que incluem uma série de estruturas em formas de agulha que detectam o movimento, gerando leituras que são transmitidas para o circuito principal. O acelerômetro tri-axial é um sensor que retorna um valor real estimado na aceleração ao longo dos eixos X, Y e Z, na qual a velocidade e deslocamento podem ser estimados, como descrito por Kaghyan [3].

O uso dessa tecnologia tem crescido principalmente devido ao fato do mercado de *smartphones* também estar crescendo. Isso passou a acontecer após o lançamento de *smartphones* que utilizam sistemas operacionais modernos como Android e iOS. Em particular, por ter código aberto e possibilidade de customização e facilidade no desenvolvimento de aplicações, muitos fabricantes aderiram ao Android. A maior característica desse sistema é que o desenvolvedor não precisa criar um aplicativo para cada tipo de celular ou *tablet*, pois ele irá funcionar em outros aparelhos que possuam o mesmo sistema operacional.

Já pensando nesse mercado a empresa *Unity Tecnologies* passou também a integrar em suas opções de plataformas o desenvolvimento para aplicativos Android, na terceira versão da *engine Unity*.

A *engine Unity* possui como uma de suas principais características, a habilidade de dar suporte a uma grande área de dispositivos móveis, mesmo eles possuindo formas de processamento gráficos completamente diferentes. Como descrito no artigo *Unity* por Zioma [1], o desenvolvimento para algumas dessas plataformas pode apresentar certas dificuldades, como a falta de ferramentas de desenvolvimento para a plataforma iOS.

O objetivo geral desse trabalho é explorar o uso de sensores em dispositivos móveis no contexto de jogos digitais, desenvolvidos em conjunto com a *engine Unity*.

Como objetivos específicos, destacam-se:

- Explorar os recursos disponíveis na *engine Unity*;
- Explorar o uso do acelerômetro na plataforma Android;
- Utilizar o acelerômetro como forma de aprendizado;
- Implementar um estudo de caso integrando a plataforma Android com a *engine Unit* para uma aplicação com finalidade de ensino.

O restante da monografia está organizada como explicado a seguir. No Capítulo 1 será abordada a visão geral e o histórico da plataforma Android. Também será detalhada a arquitetura do mesmo, incluindo informações sobre a comunicação, armazenamento de dados e outras características gerais, como interfaces gráficas. Para finalizar, será abordado sobre o emulador da plataforma para distribuição de aplicativos. O Capítulo 2 será abordado sobre o funcionamento da *engine Unity*. Também será mencionado o funcionamento dessa *engine* em conjunto com a plataforma Android. No Capítulo 3 será feito um estudo de caso, de como será o funcionamento do acelerômetro do Android ao se criar um aplicativo para o mesmo com o uso da *Unity.* Por fim, o trabalho se encerra com as conclusões e sugestões de trabalhos futuros.

1. Android

Neste capítulo será apresentada uma visão geral da plataforma Android bem como sua arquitetura, características gerais e a distribuição dos aplicativos.

1.1 Visão Geral e histórico da plataforma

Android é uma plataforma para dispositivos móveis O*pen Source* com seu sistema operacional baseado em Linux, que teve como responsáveis por cuidar de seu projeto, a *Google* em parceria com o *Open Handset Alliance*, que corresponde a mais de quarenta empresas do ramo de tecnologia móvel, líderes de dispositivos móveis e tecnologia que compartilham essa visão de mudar a experiência de consumidores desses dispositivos (BRÄHLER, 2010). Segundo Burnette (2009), essa nova plataforma está sendo usada em milhões de tipos de celulares e em diversos dispositivos móveis, tornando o Android o principal sistema operacional para os desenvolvimento de aplicativos neste tipo de ambiente**.**

Na década de 90, as crianças cresceram usando *Nintendo Game Boys* e *Sega Game Gears*, exemplos comuns de consoles portáteis da época. Elas passaram horas tentando alcançar os objetivos dos jogos ou conseguir a maior pontuação neles. Essas crianças levavam tais *hardwares* para todos os lugares que podiam. Com o passar do tempo, essa paixão por jogos fizeram-nas terem vontade de criar seus próprios jogos, seus próprios mundos para compartilhar com seus amigos (ZECHNER, 2012).

Mais tarde, esses usuários começaram a programar nos PC's, mas logo notaram que não poderiam transferir suas criações para as plataformas portáteis disponíveis. Logo, mesmo continuando entusiásticos programadores, com o tempo, o interesse dos mesmos em jogar vídeo games foi se esvaindo.

Com o passar do tempo, os *smartphones* ganharam cada vez mais espaço no mercado, tornando-se a plataforma mais usada para jogos digitais, principalmente os casuais. Eles passaram, inclusive, a competir com os clássicos Nintendo *DS* e *Playstation PSP*, que são dedicados somente ao entretenimento.

Inicialmente o sistema operacional iOS pareceu uma boa plataforma para codificação de jogos, no entanto, ele apresentava suas limitações. Os sistemas *Apple* não eram abertos, e os desenvolvedores tinham que conseguir uma aprovação da *Apple* e um *Mac* para poder compartilhar seus trabalhos. Foi nesse momento que surgiu a nova plataforma móvel, o Android.

Então 2005, uma empresa chamada Android Inc., que desenvolvia um sistema operacional móvel de mesmo nome, foi comprada pelo Google. Com a popularização do uso de smartphones trazida pelo *Iphone* da *Apple*, o Google sentiu que poderia expandir seu negócio de venda de anúncios também em dispositivos móveis (PRADO, 2011).

Atualmente a plataforma Android encontra-se numa condição em que as pesquisas apontam o sistema como o futuro líder de vendas no mercado de *tablets*, superando outros dispositivos como o *IPad* da *Apple.* O uso do Android, diferente dos sistemas da *Apple* que só são usados nos dispositivos da mesma, não se limita a um único aparelho. Diversas marcas utilizam o sistema Android em seus novos *smartphones* e *tablets*, logo, isso torna um fator bem importante na difusão da plataforma.·.

1.2 Arquitetura da plataforma

A arquitetura do sistema operacional Android é dividida em camadas, onde cada parte é responsável por gerenciar as suas respectivas atividades (LECHETA, 2010). Existem quatro delas, conforme ilustrado na Figura 1: Aplicações, Bibliotecas, *Runtime* e *Kernel Linux*.

A camada de aplicações (*Applications*) é onde estão localizados todos os aplicativos que são executados no sistema operacional, como envio de mensagens, e-mails, navegadores, calculadoras, mapas e outros. É a de maior nível, e a única acessível pelo usuário;

A camada de *framework* (*Application Framework*) onde se encontram todas as *APIs* e os recursos utilizados pelos aplicativos, como classes visuais, que incluem listas, grades, caixas de texto, botões e inclusive um navegador *web* embutido, *View system* que são os componentes utilizados na construção de aplicativos, os provedores de conteúdo (*Content Provider*), responsáveis pelo compartilhamento e acesso de dados entre as aplicações, o gerenciador de conteúdo, responsável por fornecer acesso aos recursos sem código, o gerente de notificação que exibe alerta na barra de *status* e por fim, o gestor de atividade que gerencia o ciclo de vida das aplicações (FEINBUBE, 2010). É também na mesma que se encontram outros elementos, como *Location Service, Bluetooth Service, Wi-fi Service, USB Service* e *Sensor Service.*

Após as camadas citadas anteriormente, existe a de bibliotecas (Libraries), que possui as bibliotecas C/C++ utilizadas pelo sistema. Também há as bibliotecas de multimídia, visualização de camadas 2D e 3D, funções para navegadores WEB, funções para aceleradores de *hardware*, renderização 3D, funções de acesso a banco de dados SQLite, dentre outros;

Em conjunto à camada de biblioteca está a Runtime, (*Android Runtime*) responsável pela instancia da máquina virtual *Dalvik*, uma *VM* criada pela *Google* com o intuito de fazer com que os dispositivos pudessem rodar múltiplas maquinas virtuais rodando arquivos executáveis no formato *Dalvik* (*.dex*) que é otimizado para o menor uso de memória, bateria e CPU do dispositivo (EHRINGER, 2010). Essa máquina virtual é responsável pela execução de cada aplicação executada no Android e é a melhor em termos de desempenho, maior integração com a nova geração de *hardware* e projetada para executar vários processos paralelamente.

De acordo com Pereira (2009) as aplicações escritas em Java são compiladas em *bytecodes Dalvik* e executadas usando a máquina virtual *Dalvik* que é uma máquina virtual especializada desenvolvida para o uso em

dispositivos móveis, o que permite que programas sejam distribuídos em formato binário (*bytecode*) e possam rodar em qualquer dispositivo Android.

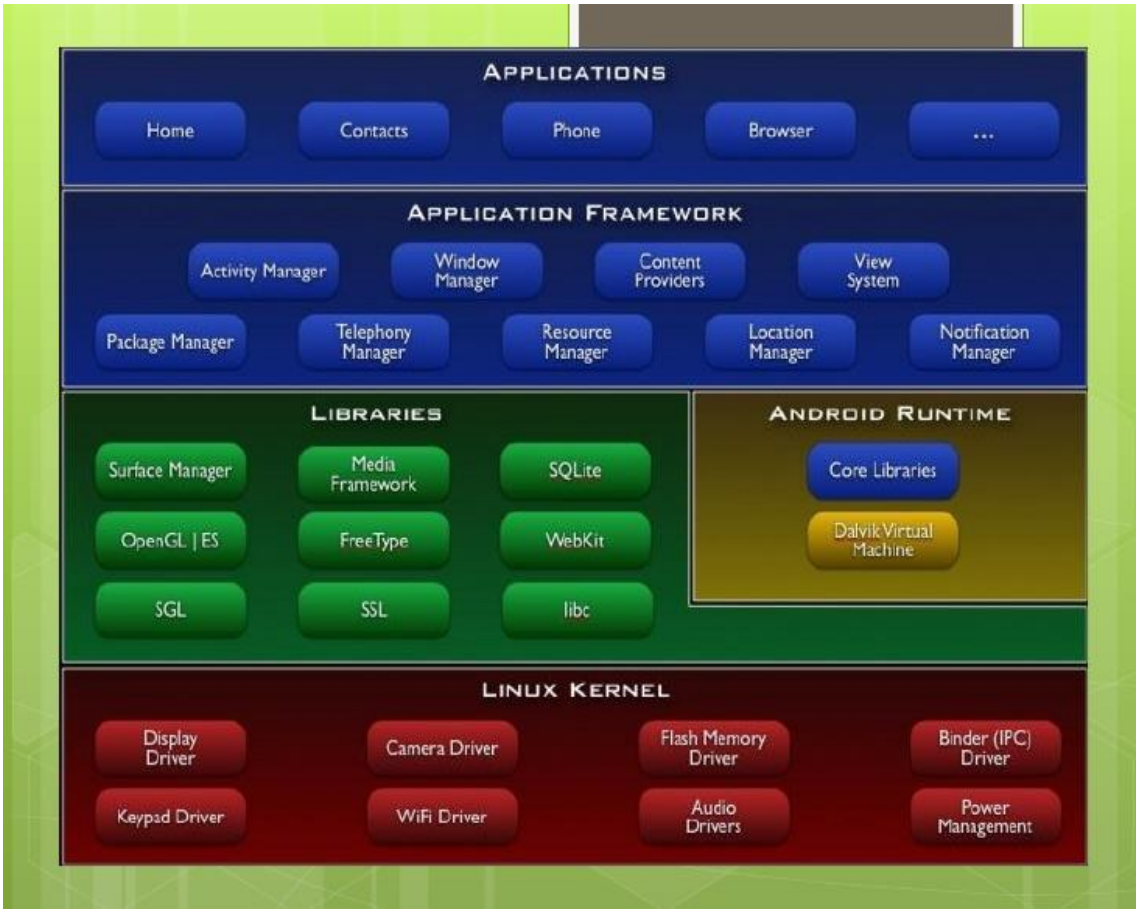


Figura 1. Arquitetura Android

Fonte: FEINBUBE (2011).

Por fim, existe a camada *Kernel Linux* (*Linux Kernel*), que possui o núcleo do sistema operacional Android onde é derivado do kernel 2.6 do Linux, herdando diversas características dessa plataforma. O motivo de usar o Linux na concepção da plataforma Android é pela forma que esse sistema operacional controla processos, gerencia memória, como faz o uso de threads, protocolos de rede, modelo de drivers e segurança dos arquivos.

Segundo Pereira (2009) nessa camada também se encontra seu próprio sistema de gerenciamento de energia, onde um aplicativo requisita o gerenciamento de energia e o driver de energia do *kernel* passa a checar periodicamente todos os dispositivos que não estão sendo utilizados por aplicações e os desliga.

1.3 Características Gerais do Sistema

1.3.1 Comunicação

O Android é uma plataforma que disponibiliza diversos meios de comunicação, como por exemplo, *bluetooth, HTTP* e o chamado NFC (*Near Field Communication*).

Um tipo de conexão sem fio usado para conectar e transmitir dados entre dispositivos em redes pessoais é o *Bluetooth*. Atualmente, diversos celulares, *smartphones*, câmeras digitais, teclados, mouses e outros dispositivos adotaram essa tecnologia por ser robusta, economizar energia e por ser de fácil implementação. No Android essa tecnologia funciona através de uma API de modo que os desenvolvedores possam utilizar esses recursos em suas aplicações. Através dela é possível ter acesso a uma pilha de protocolos *bluetooth* permitindo realizar a busca por dispositivos disponíveis em sua área de alcance, estabelecer conexão, e por fim trocar informações entre eles (MOBILE MAGAZINE, 2012).

A maioria dos aplicativos Android que utilizam algum tipo de conexão, utilizam o protocolo HTTP para enviar e receber dados. A plataforma oferece dois clientes HTTP, como o *HttpURLConnection*, que é usado para propósitos com contextos mais simples, e usado pela maioria das aplicações, e o *Apache HTTP Client*, mais usado para *web browsers* (navegadores). Os dois tipos de clientes suportam o protocolo HTTPS (que evita que as informações transmitidas entre o cliente e o servidor sejam visualizadas por terceiros), upload e download de arquivos, configurações de sessão e o protocolo IPV6.

O *NFC* é outro tipo de comunicação disponível no Android. O mesmo é um conjunto de tecnologias sem fio de curto alcance, geralmente exigindo uma distância de até quatro centímetros para que seja iniciada uma conexão. O *NFC* permite que o usuário compartilhe pequenas cargas de dados entre uma *tag NFC* (etiquetas simples programáveis com semânticas de leitura e escrita ou outras mais complexas que podem ser usadas para oferecer operações matemáticas) e um dispositivo Android, ou simplesmente entre dois dispositivos Android (DEVELOPERS).

1.3.2 Armazenamento de Dados

O sistema de banco de dados utilizado pelo Android é o SQLITE, um banco de dados *Open Source.* O SQLITE suporta padrões de bancos de dados relacionais como a sintaxe SQL, e suas operações e instruções preparadas. A utilização do SQLITE em Android não requer nenhuma configuração inicial, apenas é necessário especificar a instrução SQL para gerar o banco de dados e ele é criado automaticamente. Esse banco de dados apenas suporta dados do tipo *TEXT, INTEGER* e *REAL,* respectivamente semelhantes à *String, Long e Double* do Java. Todos os outros tipos devem ser convertidos em um desses tipos antes de armazená-los no banco de dados. O SQLITE não valida se os campos enviados para armazenamento são iguais aos campos definidos nas colunas, isso cabe ao desenvolvedor em validá-los (VOGEL, 2011).

Segundo Gonçalves (2011), o SQLITE é capaz de criar um arquivo em disco, ler e escrever diretamente sobre esse arquivo. Para ser criada uma tabela, usa-se o comando *CREATE TABLE* da linguagem SQL. E para manipular os dados dessa tabela, utilizam-se comandos DML (*INSERT, UPDATE* e *DELETE*) e suas consultas através do uso do comando *SELECT*.

1.3.3 Interfaces Gráficas

A base da interface gráfica no Android é a classe *View*. Todos os elementos visíveis da interface do usuário e alguns itens invisíveis são derivados dessa classe. O Android fornece elementos de interfaces gráficas (*UI*) padrões, como botões, texto e visualização de vídeos ou ate mesmo um selecionador de datas (BRAHLER, 2010).

A fim de interagir com a *UI*, a classe *View* fornece alguns métodos padrões como *onClick(), onTouch()* e *onKey()*, que são chamados por *frameworks* subjacentes. Em adição aos elementos da *UI*, o *framework* fornece menus e mensagens (*caixa de diálogo)*. Menus podem ser tanto de opção que podem ser acessados via um botão de atalho para o mesmo, ou menus de uma *view*. E para informar o usuário ou para obter um comando, a aplicação pode apresentar uma *dialog*. Android fornece *dialogs* para seleção de datas e horas, notificações em barras de progressão e até um *dialog* personalizado com botões.

1.3.4 Segurança

A plataforma Android foi feita para ser completamente aberta. As aplicações do mesmo fazem uso de avançados *softwares* e *hardwares*, bem como dados locais, expostos através da plataforma para trazer inovação e valor aos consumidores. Para proteger esse valor, a plataforma deve oferecer um ambiente de aplicação que garanta a segurança dos usuários.

Logo, para proteger um sistema aberto, isso exige uma arquitetura de segurança robusta, bem como programas de segurança rigorosos. A plataforma foi planejada pensando nos desenvolvedores. Controles de segurança foram projetados para reduzir a carga sobre os mesmos. Nesse sentido, programadores de segurança experientes podem trabalhar e confiar nesses flexíveis controles de segurança. Aqueles que são menos familiarizados com o desenvolvimento em segurança serão protegidos pela segurança padrão do sistema.

No início do desenvolvimento do sistema, a equipe de desenvolvimento do núcleo do Android notou que seria necessário um robusto sistema de segurança para permitir uma diversidade de aplicativos e dispositivos construídos sobre e ao redor da plataforma e que fossem apoiados por serviços de nuvem (*cloud services*). Como resultado disso, durante o seu ciclo de vida de desenvolvimento, o Android foi submetido a um programa de segurança profissional. Essa equipe observou como outros dispositivos móveis, computadores de mesa e plataformas de servidores reagiam em questões de segurança e construíram um programa de segurança para lidar com os pontos fracos observados nos testes desses outros dispositivos.

O Android possui uma arquitetura de segurança que tem como objetivo, proteger os dados do usuário, proteger os recursos do sistema (incluindo a rede) e fornecer isolamento de aplicativos (Android Open Source Project, 2010). E para isso a plataforma possui como características-chave da arquitetura de segurança:

- Segurança robusta no nível do sistema operacional através do *kernel Linux.* Essa característica, através do *Application Sandbox,* garante que nenhuma aplicação, ou código executável de uma aplicação ameace o funcionamento de outros programas, bem como o sistema operacional ou o próprio dispositivo.
- O *"Application Sandbox"* funciona como o sistema de usuários da plataforma Linux, em que os dados de um usuário não podem ser acessados por outros. Mas no caso do Android, cria-se uma "*unique user ID"* (uma identificação única) onde cada aplicação recebe um ID para que cada uma funcione em processos diferentes.
- Partições do sistema e modo de segurança que garante que, se o usuário ligar o dispositivo no modo de segurança, só aplicações nativas do Android estarão disponíveis e que aplicações de outras empresas não serão executadas no mesmo ambiente.
- O Android pode ser configurado para verificar uma senha definida pelo usuário para que o mesmo tenha acesso ao dispositivo.
- O funcionamento do Android em relação à aplicações *third-party* (aplicações de outras empresas) deixa claro para os usuários que eles estão interagindo com programas de terceiros e informa aos mesmos dos recursos que esses aplicativos têm. Antes da instalação de qualquer aplicativo, é mostrada para o usuário uma mensagem sobre as permissões que o aplicativo está solicitando, como ilustrado na Figura 2.

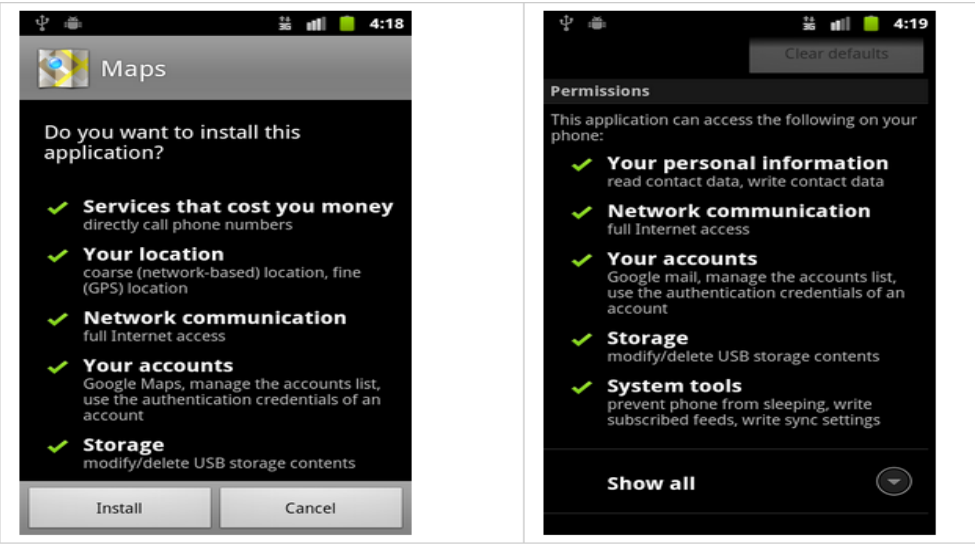


Figura 2. Solicitação de permissão para instalação de um aplicativo.

1.4 Distribuição Android

1.4.1 O Emulador

O emulador Android é uma aplicação que fornece um dispositivo móvel virtual em que o desenvolvedor pode executar seus aplicativos para Android. Ele executa uma pilha completa do sistema Android ate o nível *kernel*, que inclui uma série de aplicações pré-instaladas (por exemplo, um discador) que o desenvolvedor pode acessar por suas aplicações. O desenvolvedor pode configurar, usando as configurações do *Android Virtual Device (AVD)*, que versão do sistema Android usar para rodar o emulador. Além disso, ele também pode personalizar a aparência do dispositivo virtual e o mapeamento dos botões.

As imagens do sistema disponíveis através do gerenciador do Android SDK contém código para o *kernel Linux* do Android, as bibliotecas nativas, a máquina virtual *Dalvik*, e vários outros pacotes do sistema (como o Android *framework* e as aplicações pré-instaladas). O emulador fornece uma tradução dinâmica do código da máquina do dispositivo para o sistema operacional e para a arquitetura do processador da sua máquina de desenvolvimento.

Figura 4: Emulador Android

Fonte: DEVELOPERS

O emulador Android oferece uma grande parte das ações que o usuário pode fazer num dispositivo Android físico. Nesse emulador, como mostrado na Figura 4, o desenvolvedor pode emulador características nativas como: um cartão de memória, emular conexões de rede, simular o envio de mensagens

SMS e alguns tipos de aplicativos desenvolvidos pelo desenvolvedor. Por outro lado, existem algumas características de um Android em um dispositivo físico que o emulador não pode emular, como: conexões USB, o uso de fones de ouvido, não pode simular *bluetooth*, incapaz de simular a condição da bateria, dentre algumas outras.

1.4.2 "Build"

Alguns dos requisitos para um ambiente de desenvolvimento Android são determinados por qual versão do código fonte o desenvolvedor planeja usar para compilar seu projeto. Cada dispositivo móvel que possui a plataforma tem uma versão em que o sistema operacional será executado. Ao escolher para qual dispositivo será desenvolvida a aplicação, o desenvolvedor pode escolher qualquer uma das versões disponíveis para o mesmo.

O ambiente operacional do Android é baseado no *kernel* Linux, mas por utilizar a linguagem JAVA para a codificação, isso torna viável para o desenvolvimento de aplicativos da plataforma, a configuração do ambiente em qualquer sistema operacional. Contanto que o desenvolvedor tenha feito a instalação da JDK (conjunto de bibliotecas de desenvolvimento Java), configurado o Android SDK (kit de desenvolvimento de software, incluindo o emulador e as bibliotecas) para que seja usado em um ambiente de desenvolvimento, como a *IDE Eclipse*, o programador agora pode começar o desenvolvimento de aplicações para a plataforma (IBM, 2009).

Ao se criar um projeto, as ferramentas da SDK esperam que o desenvolvedor siga uma estrutura específica para o desenvolvimento do aplicativo para que o programa seja compilado e empacotado corretamente. A própria Android SDK fornece *templates* para a rápida criação de uma aplicação com uma básica estrutura ou para simplesmente adicionar componentes para um projeto já existente. Os códigos dos *templates* fornecidos seguem as diretrizes de desenvolvimento e *design* para direcionar corretamente o desenvolvedor para criar uma aplicação bonita e funcional.

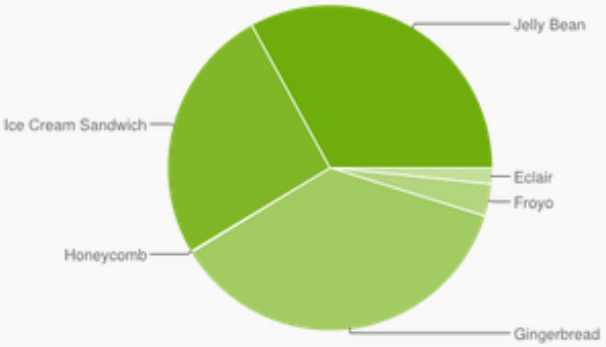
1.4.3 Distribuição

O Android não está ligado a um *hardware*, ou seja, a um único aparelho. Existem muitas variáveis que apenas o *hardware* para a distribuição da plataforma nos mesmos. Para cada aparelho, há versões da plataforma compatíveis com o mesmo e cada uma delas tem suas diferenças e atualizações de uma para outra. A seguir elas serão citadas bem como suas características, e um gráfico (Figura 3) apontando a distribuição dessas versões nos aparelhos em sua totalidade pelo mundo.

Segundo Brahler (2010) a primeira versão, *Cupcake* (1.5), baseada no *kernel Linux* 2.6, foi a primeira atualização do sistema que havia sido lançado (Android 1.1), oferecendo novas interfaces de usuário, aplicações baseadas em acelerômetro, gravação e execução de vídeos, *bluetooth* e uma interface de teclado na tela. Logo após, no mesmo ano foi lançada a versão *Donut* (1.6).

Com ela veio o suporte a maiores resoluções na tela, uma *engine* criada para executar fala através de textos digitados e a criação de uma rede privada virtual. O Android 2.1 *Eclair* foi a primeira versão que fez diferença nos dispositivos móveis, apresentando uma grande variedade de aplicativos do Google, como o Gmail, por exemplo, bem como uma atualização robusta na interface do usuário. Em seguida o Android 2.2 *Froyo* que teve como principais características, otimização de desempenho e suporte para o *Adobe Flash* e para *OpenGL.*

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.5%
2.2	Froyo	8	3.2%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	36.4%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	25.6%
4.1.x	Jelly Bean	16	29.0%
4.2.x		17	4.0%



Data collected during a 14-day period ending on June 3, 2013.
Any versions with less than 0.1% distribution are not shown.

Figura 3. Gráfico de distribuição de versões.

Fonte: Android Open Source Project.

No Android *Gingerbread* (2.3), iniciou-se o suporte a tecnologias como o NFC, sensores de movimento para jogos, chamadas por VOIP (voz sobre IP), além de uma interface mais funcional. Estabilizou-se como o sistema mais utilizado do mercado. A versão 4.0, a *Ice Cream Sandwich,* que teve como principal característica, uma maior eficiência da tecnologia NFC. Também nessa versão foi implementado um sistema de reconhecimento facial no momento de desbloquear o dispositivo. E o mais novo sistema do Android, agora a *Jelly Bean* (4.1 e 4.2) uma versão mais rápida e que consome menos bateria, tem como principal característica a otimização das principais ferramentas do *smartphone*, bem como a opção de redimensionar os ícones dos aplicativos.

Ao preparar uma aplicação para ser lançada, primeiramente o desenvolvedor configura, constrói seu código e testa a versão de publicação da aplicação. A parte da configuração é direta, envolve codificação básica de limpeza e modificações para ajudar a otimizar a aplicação. O processo de construção é similar ao depurar do código durante o desenvolvimento e pode ser feita via JDK ou pela própria ferramenta da Android SDK. As tarefas de teste servem como uma confirmação final, garantindo que a aplicação se

comporta como o esperado em condições reais. Quando termina de preparar a aplicação para publicação, gera-se um arquivo "*.apk"* que podem ser distribuídas diretamente para usuários ou através de uma aplicação de mercado, como por exemplo, o *Google Play*.

Referências Bibliográficas

[1] BLACKMAN, Sue. Beginning 3D Game Development with Unity: All-in-one, multi-platform game development. : Apress, 2011.

[2] BRAHLER, S. (2010). Analysis of the Android architecture.

[3] BURNETTE, Ed. Hello, Android: Introducing Google's Mobile Development Platform.

[4] EHRINGER, David; The Dalvik Virtual Machine Architecture. 2010

[5] FEINBUBE, Frank; POLZE, Andreas: Embedded OS - Android | FF | 17. November 2011.

[6] FIGUEIREDO, Luiz Carlos. Modelagem Matemática de sistemas de rastreamento de veículos. Ipatinga: Doxa, 2002.

[7] GONÇALVES, Eduardo Corrêa. SQLite, Muito Prazer!. Disponível em

<http://www.devmedia.com.br/post-7100-SQLite-Muito-Prazer.html>. Ultimo acesso em 20/05/2013.

[8] KAGHYAN, Sahak; SARUKHANYAN, Hakob. Activity recognition using K-nearest neighbor algorithm on smartphone with tri-axil accelerometer. : International Journal "Information Models and Analyses", 2012.

[9] LECHETA, Ricardo. Google Android - Aprenda a criar aplicações para dispositivos móveis com Android SDK. 2. Ed. São Paulo: Novatec Editora, 2010.

[10] PARIKH, Neel. Accelerometer based motion gestures for Mobile Devices. Master's Projects, Paper 103, 2008. Disponível em: <http://scholarworks.sjsu.edu/etd_projects/103>. Ultimo acesso em 30/03/12.

[11] PEREIRA, Lucio; SILVA, Michel: Android para desenvolvedores.: Brasport, 2009.

[12] PRADO, Sergio; Introdução ao funcionamento interno do Android, 2011. Disponível em: <http://sergioprado.org/introducao-ao-funcionamento-interno-do-android/>. Último acesso em 22/05/2013

[13] P.Rogers, Michael. Bringing Unity to the Classroom: 2012.

[14] VOGEL, Lars. Android SQLite Database - Tutorial. 2011. Disponível em

<http://www.vogella.de/articles/AndroidSQLite/article.html>. Último acesso em 20/05/2013

[15] ZECHNER, Mario; GREEN, Robert. Beginning Android 4 Games Development. : Apress, 2012.

[16] Zioma, Renaldas. Unity: IOS and Android – Cross Platform Challenges and Solutions, Los Angeles, 2012.

[17] Android Open Source Project: Android Security Overview.:https://source.android.com/tech/security/#android-security-overview. Ultimo acesso em 25/05/2013.

[18] Comunicação via Bluetooth no Android.: Mobile Magazine, 2012.: http://www.devmedia.com.br/comunicacao-via-bluetooth-no-android-artigo-webmobile-35/20464. Ultimo acesso em 25/05/2013.

[19] Near Field Communication.: http://developer.android.com/guide/topics/connectivity/nfc/index.html. Ultimo acesso em 27/05/2013.

[20] Introdução ao desenvolvimento do Android, 2009.: http://www.ibm.com/developerworks/br/library/os-android-devel/. Ultimo acesso em 28/05/2013.