

Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Curso Bacharelado em Sistemas de Informação

Ana Carine Almeida Nascimento

**Levantamento e Análise dos Recursos Computacionais de
Dispositivos Móveis *Android* Utilizados em Jogos**

Picos
2014

Ana Carine Almeida Nascimento

Levantamento e Análise dos Recursos Computacionais de Dispositivos Móveis *Android*
Utilizados em Jogos

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharelado, sob orientação da Professora Juliana Oliveira de Carvalho.

Picos
2014

Eu, **Ana Carine Almeida Nascimento**, abaixo identificado(a) como autor(a), autorizo a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação abaixo discriminada, de minha autoria, em seu site, em formato PDF, para fins de leitura e/ou impressão, a partir da data de hoje.

Picos-PI 12 de março de 2014.

Ana Carine Almeida Nascimento
Assinatura

FICHA CATALOGRÁFICA

Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

N244I Nascimento, Ana Carine Almeida.
Levantamento e análise dos recursos computacionais de dispositivos móveis Android utilizados em jogos / Ana Carine Almeida Nascimento. – 2013.
CD-ROM : il. ; 4 ¾ pol. (47 p.)

Monografia(Bacharelado em Sistemas de Informação) –
Universidade Federal do Piauí. Picos-PI, 2013.
Orientador(A): Profa. MSc. Juliana Oliveira de Carvalho

1. Dispositivos Móveis. 2. Android. 3. Recursos Computacionais. 4. Jogos I. Título.

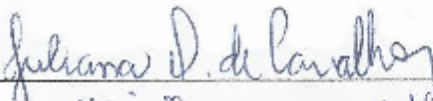
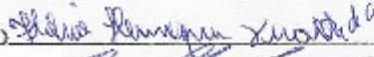

CDD 005.26

Ana Carine Almeida Nascimento

Levantamento e Análise dos Recursos Computacionais de Dispositivos Móveis *Android*
Utilizados em Jogos

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharelado, sob orientação da Professora Juliana Oliveira de Carvalho.

Data de Aprovação: 30/03/2014

Juliana Oliveira de Carvalho  UFPI - CSHNB
Flavio Henrique Duarte de Araujo  UFPI - CSHNB
Ismael Holanda Leal  UFPI - CSHNB

Picos
2014

Dedico esse trabalho primeiramente ao meu Deus, pois sem Ele seria impossível minha existência.

Dedico aos meus pais, que sempre me apoiaram e são meu exemplo de vida e a minha irmã que para mim também é uma guerreira, amo vocês. Por fim, dedico esse trabalho a todos meus amigos e irmãos que sempre estiveram do meu lado me ajudando e intercedendo por mim.

Porque Dele por Ele, para Ele são todas as coisas. Depois de vários anos, hoje posso expressar minha enorme gratidão ao meu Deus, razão da minha vida.

Agradeço aos meus pais, Rui Nascimento e Socorro Nascimento pelo apoio, força e carinho que sempre estiveram dispostos a me dar, vocês são meus exemplos de vida e amores da minha vida. À minha irmã, Ana Caroline que tanto amo e que com sua garra e determinação me inspira a lutar sempre.

À todos professores que ao longo desse curso contribuíram para minha formação. Em especial a Prof^a. Ms. Juliana Oliveira de Carvalho, exemplo de mulher e professora, pela qual tenho enorme carinho, admiração e gratidão pelo apoio e paciência nesse trabalho e nas demais disciplinas ao longo desses anos. À doce Prof^a. Ms Patrícia Medyna agradeço pela disponibilidade e apoio.

À todos os meus amigos que me ajudaram e estiveram do meu lado nessa caminhada, Tia Lucineide, Karolyne, Camila, Higor, Luciana, Fátima, Mirielly, Andrey, George, Pablo que apesar da distância sempre esteve presente, Woshington, Cliciano, Taynara, Bruna Pamera, à pequena notável Danila que tanto me ajudou, sem vocês com certeza seria menos colorida a vida.

À todos que contribuirão direta ou indiretamente para realização desse trabalho, muito Obrigado!

"Bendize, ó minha alma, ao Senhor, e não te esqueças de nenhum dos seus benefícios."

(Salmos 103:2)

"Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível."

(Charles Chaplin)

Resumo

Com a evolução da *Internet*, houve o crescimento em larga escala do uso de dispositivos móveis e mais especificamente de *smartphones*, sendo o *Android* a plataforma com maior número de usuários. É crescente a quantidade de aplicações principalmente jogos, que são desenvolvidos para a plataforma *Android*, como são aplicações que consomem muitos recursos computacionais, este trabalho se propõe a analisar quais recursos são mais utilizados no momento da execução, através da realização de testes com uma variedade de jogos, em diversos aparelhos com características físicas diferentes. Os resultados obtidos poderão auxiliar os desenvolvedores na otimização do desenvolvimento de jogos para *Android* e também ajudar os usuários a escolherem o melhor dispositivo de acordo com suas necessidades.

Palavras-chave: Dispositivos Móveis Android. Recursos Computacionais. Jogos.

Abstract

With the evolution of the Internet, there was a large scale growth of the use of mobile devices, more specifically of smartphones, with Android platform holding the largest number of users. There is a growing number of applications, mainly games that are developed for the Android platform, are applications that consume much computational resources, this study aims to examine which resources are more used at run time, by conducting tests with a variety of games on various devices with different physical characteristics The results may assist game developers in optimizing the development of games for Android and also help users to choose the best device to suit their needs.

Keywords: Android Mobile devices. Computational resources. Games.

-: ||||| -:

Lista de Figuras

Figura 1 -	<i>Ciclo de Vida de uma Activity</i>	22
Figura 2 -	<i>DDMS- Uso de Memória</i>	28
Figura 3 -	<i>QtADB</i>	29
Figura 4 -	<i>Quadrant Standard</i>	30
Figura 5 -	<i>Angry Birds Rio</i>	32
Figura 6 -	<i>SpeedCar</i>	33
Figura 7 -	<i>Hill Climb Racing</i>	33
Figura 8 -	<i>Tela Inicial do Jetpack Joyride</i>	34
Figura 9 -	<i>Game Fruit Ninja</i>	34

Lista de Tabelas

Tabela 1 -	Métodos do ciclo de vida de uma aplicação	23
Tabela 2 -	Dispositivos Utilizados na Execução dos Testes	31
Tabela 3 -	Ferramenta – DDMS <i>game Angry Birds</i>	37
Tabela 4 -	Ferramenta – <i>Quadrant Standard game Angry birds</i>	38
Tabela 5 -	Ferramenta – DDMS <i>game Speed car</i>	39
Tabela 6 -	Ferramenta – <i>Quadrant Standard game Speed Car</i>	40
Tabela 7 -	Ferramenta – DDMS <i>game Hill Climb Racing</i>	40
Tabela 8 -	Ferramenta – <i>Quadrant Standard game Hill Climb Racing</i>	41
Tabela 9 -	Ferramenta – DDMS <i>game Jetpack Joyride</i>	42
Tabela 10 -	Ferramenta – <i>Quadrant Standard game Jetpack Joyride</i>	43
Tabela 11 -	Ferramenta – DDMS <i>game Fruit Ninja</i>	43
Tabela 12 -	Ferramenta – <i>Quadrant game Standart Fruit Ninja</i>	44

Lista de abreviaturas e siglas

AAC	Advanced Audio Coding
ABINEE	Associação Brasileira da Indústria Elétrica e Eletrônica
ADB	Android Debug Bridge
AMOLED	Active-Matrix Organic Light-Emitting Diode
AMR	Automatic meter reading
CPU	Central Processing Unit
DDMS	Dalvik Debug Monitor Server
dp	Density-Independent Pixel
DVM	Máquina Virtual Dalvik
GC	Garbage Collector
GHz	Gigahertz
I/O	Entrada e saída
IDC	International Data Corporation
JPG	Joint Photographic Experts Group
LRU	Least Recently Use
MB	Megabyte
Mhz	Megahertz
MMU	Memory Management Unit
MP3	Media Player
MPEG4	Moving Picture Experts Group
OLED	Organic Light-Emitting Diode
OOM	Out-of-Memory Handler
PNG	Portable Network Graphics
RAM	Random-Access Memory
SD	Secure Digital Card
SDK	Software Development Kit
TFT	Thin Film Transistor
UI	User Interface
USB	Universal Serial Bus

Sumário

1	Introdução	15
2	Disponibilidade de Recursos Computacionais para Dispositivos Móveis com <i>Android</i>	17
2.1	Memória RAM (<i>Random-Access Memory</i> - Memória de Acesso Aleatório) . . .	17
2.1.1	Gerenciamento de Memória	18
2.1.2	Memória para dispositivos móveis	18
2.1.3	Gerenciamento de Memória no <i>Android</i>	18
2.2	Processador	19
2.2.1	Gerência do Processador	20
2.2.2	Gerenciamento de Processos no <i>Android</i>	21
2.2.3	<i>Threads</i>	23
2.3	Importância da Disponibilidade de Recursos Computacionais para <i>Android</i> na Execução de <i>Games</i>	24
2.3.1	<i>Cache</i> no <i>Android</i>	24
2.3.2	Tamanho de Tela	24
2.3.3	Bibliotecas	25
3	Descrição de Ferramentas de Análise e Teste de Desempenho, Dispositivos e <i>Games</i>	27
3.1	DDMS (<i>Dalvik Debug Monitor Server</i>)	27
3.2	QtADB	28
3.3	<i>Quadrant Standard</i>	29
3.4	Especificações dos Dispositivos Utilizados na Realização dos Testes	30

3.5	Descrição dos Jogos	31
3.5.1	<i>Angry Birds</i> Rio	32
3.5.2	<i>SpeedCar</i>	32
3.5.3	<i>Hill Climb Racing</i>	33
3.5.4	<i>Jetpack Joyride</i>	34
3.5.5	<i>Fruit Ninja</i>	34
4	Resultados e Discussões	36
4.1	Testes de Execução	36
4.1.1	Execução com Ferramenta DDMS	36
4.1.2	Execução com Ferramenta <i>Quadrant Standard</i>	37
4.2	<i>Game Angry Birds</i>	37
4.2.1	Execução com Ferramenta DDMS	37
4.2.2	Ferramenta <i>Quadrant Standard</i>	38
4.3	Testes na execução do <i>game Speed car</i>	39
4.3.1	Execução com Ferramenta DDMS	39
4.3.2	Ferramenta <i>Quadrant Standard</i>	40
4.4	Testes na execução do <i>game Hill Climb Racing</i>	40
4.4.1	Execução com Ferramenta DDMS	40
4.4.2	Ferramenta – <i>Quadrant Standard Hill Climb Racing</i>	41
4.5	Testes na execução do <i>game Jetpack Joyride</i>	42
4.5.1	Execução com Ferramenta DDMS	42
4.5.2	Ferramenta <i>Quadrant Standard</i>	43
4.6	Testes na execução do <i>game Fruit Ninja</i>	43
4.6.1	Execução com Ferramenta DDMS	43
4.6.2	Ferramenta <i>Quadrant Standard</i>	44
4.7	Análise Geral da Ferramenta DDMS	44

4.8	Análise Geral da Ferramenta <i>Quadrant Standard</i>	45
5	Conclusão	46
	Referências	47

1 Introdução

Com a ascensão do uso dos dispositivos móveis, houve um crescimento na venda de *smartphones* em todo o planeta. A plataforma móvel *Android* é sem dúvidas a que possui o maior número de usuários (ANDROID, 2012), além do número de aplicações que são desenvolvidas para a plataforma que também é crescente. O sistema operacional *Android* disponibiliza muitos recursos, que ao serem executados juntamente com o *hardware* dos *smartphones* que estão disponíveis no mercado atual, proporcionam aos usuários recursos diversos na execução de *games*. Esse fato é de grande importância também para os desenvolvedores.

Os jogos para *smartphones* são um dos mais importantes entretenimento digital, e com o avanço tecnológico dos celulares são desenvolvidas aplicações mais elaboradas, tanto na qualidade dos efeitos gráficos quanto nos sonoros e com isso possibilitar o surgimento de um mercado importante, pois *smartphones* de última geração permitem a instalação de muitos jogos e aplicativos através de lojas *online* (MORIBE, 2012).

Segundo dados da IDC (*International Data Corporation*) divulgados pela Associação Brasileira da Indústria Elétrica e Eletrônica (ABINEE), mostram que a venda de *smartphones* cresceram 85%, chegando a 5,4 milhões de unidades no primeiro trimestre de 2013, em relação ao mesmo período de 2012 (ABINEE, 2013). E em relação ao uso de jogos em *smartphones*, uma pesquisa recente divulgada pela empresa de inteligência de mercado *Juniper Research*, mostrou que *smartphones* e *tablets* serão os dispositivos primários para os *games*. A empresa projeta que 64,1 bilhões de *downloads* de aplicativos de jogos para dispositivos móveis sejam realizados em 2017, em comparação a 21 bilhões em 2012. Isso acontece pelo fato da memória dos dispositivos móveis aumentarem cada vez mais sua capacidade de armazenamento, oferecendo mais espaço para os jogos (RESEARCH, 2013).

Um fator de grande importância, é a boa performance dos recursos computacionais em dispositivos móveis, principalmente quando se trata da execução de *games*. Por esse motivo os desenvolvedores de *games*, não podem deixar de observar a capacidade dos recursos que irão executar as aplicações desenvolvidas, além de analisar as medidas de desempenho do *game*, quais recursos computacionais e com que frequência são alocados, e ainda observar o tempo de execução. Portanto, esse trabalho se propõe em obter informações relacionadas ao desempenho dos jogos e sobre o consumo de recursos na execução de *games* com sistema operacional *Android*. Os testes serão realizados com auxílio de ferramentas de análise e teste de desempenho,

dentre elas DDMS e *Quadrant Standard*.

Desta forma a monografia será dividida em 5 capítulos dos quais são apresentados a seguir em um breve resumo:

- Capítulo 2 - Será abordado sobre o consumo e a importância da disponibilidade de Recursos Computacionais para dispositivos móveis com sistema operacional *Android*, tais como memória, processador, gráficos 2D e 3D e tamanho de tela.
- Capítulo 3 - Apresentação e descrição das ferramentas de análise e teste de desempenho que serão utilizadas, além da descrição dos dispositivos móveis, emuladores e jogos que participarão dos testes.
- Capítulo 4 - Descrição dos resultados obtidos nos testes. Descrever cada resultado individual obtido nos testes, em relação a ferramenta, dispositivo e jogo executado. Em seguida realizar uma comparação e análise dos dados obtidos.
- Capítulo 5 - O capítulo da conclusão mostra quais objetivos realmente foram alcançados, quais contribuições o trabalho trouxe para desenvolvedores e usuários de *games* em *Android*.

2 Disponibilidade de Recursos Computacionais para Dispositivos Móveis com *Android*

Os avanços tecnológicos ocorridos nos últimos anos proporcionou-se o surgimento de dispositivos móveis mais potentes e com novas funcionalidades, quando comparados aos *desktops* e *notebooks*, ainda possuem relativamente algumas desvantagens como: menor poder de processamento, memória RAM limitada, capacidade de armazenamento persistente restrita, Tela pequena e de baixa resolução, altos custos associados à transferência de dados, conexão pouco confiável e de baixa velocidade e tempo de vida de bateria curta. Portanto, estas características devem ser consideradas durante o processo de desenvolvimento de aplicações para dispositivos móveis em geral (MARTINS, 2009).

Este capítulo irá abordar o conceito de cada recurso computacional que serão avaliados nos testes de análise e desempenho. Além de realizar uma abordagem sobre o consumo e a importância da disponibilidade de recursos computacionais para dispositivos móveis com sistema operacional *Android*, tais como memória, processador, gráficos 2D e 3D e tamanho de tela.

2.1 Memória RAM (*Random-Access Memory* - Memória de Acesso Aleatório)

A memória RAM é um dos componentes de grande importância na arquitetura computacional. Uma característica distinta da RAM é a possibilidade de ler dados da memória e escrever novos dados de maneira mais rápida. Tanto a leitura quanto a escrita são realizadas por meio de sinais elétricos. Uma outra característica fundamental é o fato de ela ser volátil, precisa receber uma fonte de alimentação constante, se a energia for interrompida os dados são perdidos. É uma memória de acesso aleatório, ou seja, palavras individuais da memória são acessadas diretamente por meio da lógica de endereçamento interna (STALLINGS, 2010).

2.1.1 Gerenciamento de Memória

O sistema operacional possui acesso à memória e coordena a utilização desta por processos dos usuários e garante a utilização segura da mesma. Grande parte dos sistemas operacionais utilizam o conceito de memória virtual. O sistema deve portanto assegurar que cada processo tenha seu próprio espaço na memória, prover a proteção deste espaço para que não haja uma sobrescrita e utilização por outro processo e possibilitar que uma aplicação não utilize mais memória que a existente (GOMES et al., 2012).

A técnica de gerenciamento que determinado sistema operacional utiliza depende da arquitetura do computador. A memória lógica de um processo é aquela que o processo é capaz de acessar. Os endereços manipulados pelo processo são endereços lógicos. A memória física é implementada pelos circuitos integrados de memória, pela eletrônica do computador. O endereço físico é usado para endereçar os circuitos integrados de memória. O espaço de endereçamento lógico de um processo é formado por todos os endereços lógicos que esse processo pode gerar. Existe um espaço de endereçamento lógico por processo. Já o espaço de endereçamento físico é formado por todos os endereços aceitos pelos circuitos integrados de memória. A unidade de gerência de memória MMU (*Memory Management Unit*) é o componente do *hardware* responsável por prover os mecanismos que serão usados pelo sistema operacional para gerenciar a memória (CARISSIMI et al., 2011).

2.1.2 Memória para dispositivos móveis

A capacidade de armazenamento da memória em dispositivos móveis é limitada, é importante considerar que em ambientes móveis, a otimização na utilização de espaço de memória e a eficiência no acesso aos dados são fatores que estão diretamente relacionados a um bom desempenho destes dispositivos e implicam em uma melhor performance na execução das tarefas (PITOMBEIRA, 2006).

2.1.3 Gerenciamento de Memória no *Android*

O uso de memória é geralmente limitado a 16 MB em aplicações *Android*. Essa característica se dá pela capacidade contida em um dispositivo móvel. Quanto mais aplicações o *Android* puder manter na memória, mais rápido ele será para o usuário quando este precisar trocar de aplicações. Então, as aplicações devem, preferencialmente usar o mínimo de memória possível para garantir várias aplicações rodando ao mesmo tempo sem que sejam finalizadas (HUBSCH, 2012).

Como o *Android* é um sistema totalmente multitarefa, o que permite que o usuário

navegue na *Internet* enquanto se faz um *download* e escuta uma música. O *Android* conta com gerenciamento automático de memória que é gerenciado pelo *garbage collector* (GC) utilizado para eliminar objetos que não estão sendo mais utilizados, liberando assim espaço na memória. Porém, todas as vezes que o GC é executado provoca o interrompimento por alguns milissegundos, no processamento das aplicações que estão executando no momento. Aplicações como jogos, são as que mais sofrem quando o GC está sendo executado. Para evitar travamentos na execução de aplicativos e problemas de desempenho, é importante que não sejam criados objetos desnecessários (MUNIZ, 2012).

O *Android* utiliza também o mecanismo OOM (*Out-of-Memory Handler*) para terminar processos quando ocorre falta de memória, além do *ashmem* e *pmem*. O *ashmem* é um alocador de memória compartilhada, com melhor suporte a dispositivos com pouca capacidade de memória. Já o *pmem* é um alocador de memória de processo, e é utilizado para o gerenciamento de grandes regiões contíguas de memória física compartilhadas entre o espaço dos usuários e *drivers* do *kernel* (BORDIN, 2012).

2.2 Processador

O processador é o elemento principal da arquitetura do controlador digital e tem como funções o controle dos barramentos, o gerenciamento das comunicações com a memória e os dispositivos de entrada e saída, e a execução das instruções. O processador interpreta os sinais de entrada, executa a lógica de controle segundo as instruções do programa de aplicação, realiza cálculos e executa operações lógicas, para, em seguida, enviar os sinais apropriados às saídas. Durante o ciclo de instrução, o microprocessador busca as instruções armazenadas na memória e executa cada uma delas. A execução do programa consiste na repetição sequencial do processo de busca e execução das instruções (COSTA, 2006).

A CPU é dividida internamente em duas unidades fundamentais: unidade de dados e unidade de controle. Estas unidades funcionam em conjunto e são conectadas entre si. A unidade de dados realiza cálculos aritméticos, funções lógicas, manipulação de dados, armazenamento temporário de dados, recebimento de dados e envio de dados. Enquanto que a unidade de controle é responsável pela geração dos sinais de controle da unidade de dados, geração dos sinais de controle externos, sincronização dos sinais de controle, inicialização do sistema, geração do endereço de memória, além de busca e armazenamento das instruções e dados (LACERDA, 2000).

2.2.1 Gerência do Processador

Em um sistema multiprogramado diversos programas são mantidos na memória ao mesmo tempo. Um processo é definido como um programa em execução, é um elemento ativo, que altera o seu estado, à medida que executa um programa. É o processo que faz chamadas de sistema, ao executar um programa. É possível que vários processos executem o mesmo programa ao mesmo tempo. Cada processo representa uma execução independente (CARISSIMI et al., 2011).

Processos são criados e destruídos. O momento e a forma pela qual eles são criados e destruídos depende do sistema operacional em consideração. A forma mais flexível de operação é permitir que processos possam ser criados livremente, através de chamadas de sistema. Além da chamada de sistema "cria processo", serão necessárias chamadas para "autodestruição do processo" e também para "eliminação de outro processo". A mudança de estado de qualquer processo é iniciada por um evento, que aciona o sistema operacional e então altera o estado de um ou mais processos. Uma chamada de sistema é necessariamente feita pelo processo no estado executando. Ele fica no estado bloqueado até o atendimento. Com isso, o processador fica livre. O sistema operacional então seleciona um processo da fila de aptos para receber o processador. O processo selecionado passa do estado de apto para o estado executando. O módulo do sistema operacional que faz essa seleção é chamado escalonador (CARISSIMI et al., 2011).

Segundo BESSA (2013), existem alguns fatores que afetam a velocidade de processamento, tais como:

- O tamanho dos registradores, determina a quantidade de dados com a qual o computador pode trabalhar em um dado instante.
- A quantidade de RAM também pode afetar a velocidade, pelo fato da CPU guardar uma parte maior do programa e dos dados ativos na memória, recorrendo com menos frequência ao meio do armazenamento.
- O relógio do sistema (*clock*) do computador define o ritmo da unidade central de processamento.
- A largura do barramento de dados determina quantos *bits* de cada vez podem ser transmitidos entre a CPU e outros dispositivos, além da arquitetura.
- Tamanho (ou largura) do barramento de endereços determina o número de *bytes* de memória que a CPU é capaz de acessar.

2.2.2 Gerenciamento de Processos no *Android*

Toda aplicação *Android* executa seu próprio processo, com sua própria instância da máquina virtual *Dalvik* (DVM). A DVM permite que um dispositivo possa executar múltiplas máquinas virtuais concorrentemente de maneira eficiente. Os arquivos *.class* gerados são transformados no formato *.dex* pela ferramenta *dx* e são executados pela DVM, que também usa o *kernel* do GNU/Linux para prover a funcionalidade de múltiplas *threads* e gerenciamento de memória de baixo nível (AQUINO, 2007).

O desenvolvedor não tem controle sobre as mudanças de estado de uma aplicação, isso é responsabilidade do sistema operacional, no entanto, pode ser notificado quando um estado está prestes a mudar através da chamada de um método *on[Evento]*. Sobrescrevendo esses métodos na sua classe *Activity*, o *Android* vai executá-los no momento apropriado (CASTRO; JÚNIOR, 2011).

O sistema operacional é quem decide a duração de cada aplicação, tendo como base as partes dessa aplicação que estão funcionando, a importância desse processo para o sistema e a quantidade de memória disponível. O *Android* prioriza os processos com os quais o usuário está interagindo, escolhendo descartar processos não usados pelo usuário quando ocorre falta de algum recurso. O *Android* utiliza *Activity*, que é uma aplicação em execução que exhibe algo na tela e/ou interage com o usuário. Cada *Activity* em execução possui um processo e o gerenciamento desses processos é feito por meio de uma pilha. Quando uma nova atividade é criada, ela passa a ocupar o topo da pilha, tornando-se o processo mais importante. A Figura 1 demonstra como o ciclo de vida de uma *activity* (MUNIZ, 2012).

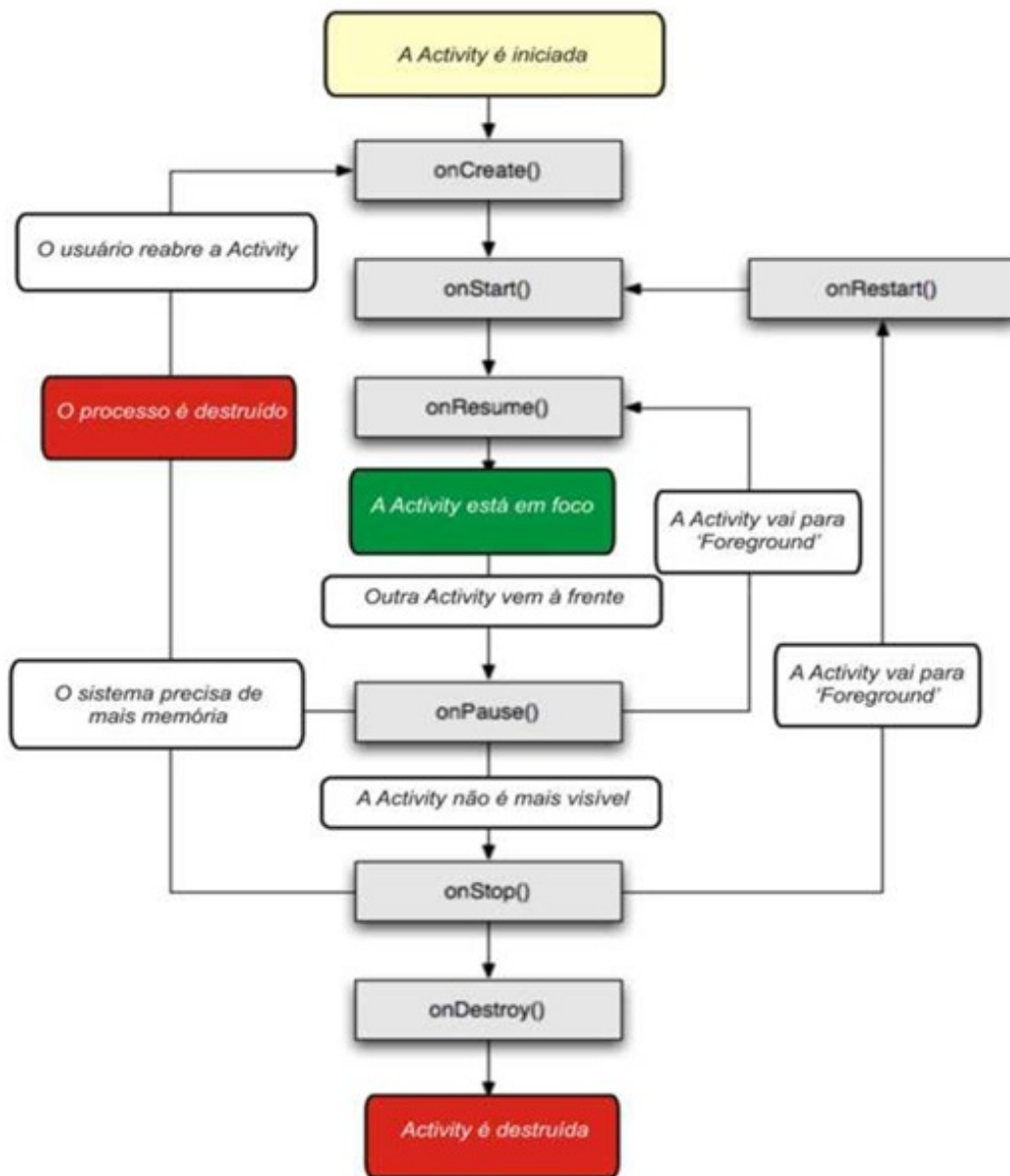


Figura 1 – Ciclo de Vida de uma Activity

Cada ciclo se inicia durante a chamada de um dos métodos controladores e termina quando algum outro método é chamado. A Tabela 1 mostra as descrições de cada um destes métodos (GONÇALVES, 2011).

Método	Descrição
<code>onCreate(bundle)</code>	Método obrigatório e invocado uma única vez, neste momento cria-se a <i>view</i> e invoca-se o método <code>setContentView(view)</code> para que seja exibida a tela representada pela <i>view</i> . Depois de finalizado, o método <code>onStart()</code> é chamado para que se dê início ao ciclo de vida visível da <i>activity</i> .
<code>onRestart()</code>	Método chamado quando a <i>activity</i> está ficando visível ao usuário, dependendo do estado da aplicação pode ser chamado depois do método <code>onCreate()</code> ou <code>onRestart()</code> .
<code>onStart()</code>	Método chamado quando uma <i>activity</i> está parada temporariamente e em processo de reinício, de forma automática chama o método <code>onStart()</code> .
<code>onResume()</code>	Método chamado quando a <i>activity</i> começa a interagir com o usuário, representa o estado de execução da aplicação. É chamado sempre depois do método <code>onStart()</code> .
<code>onPause()</code>	Método chamado quando algum evento ocorrer com o dispositivo e que faça com que a <i>activity</i> encontrada no topo da pilha tenha de ser temporariamente interrompida. É normalmente utilizado para gravar dados não salvos até o momento, para que tudo possa ser recuperado, se necessário, no método <code>onResume()</code> .
<code>onStop()</code>	Método chamado quando a <i>activity</i> está sendo encerrada, e não está mais visível ao usuário. Se a <i>activity</i> for reiniciada, o método <code>onRestart()</code> é chamado ou no caso de ficar muito tempo parada o <i>Android</i> pode chamar automaticamente o método <code>onDestroy()</code> para remover completamente a <i>activity</i> da pilha.
<code>onDestroy()</code>	Método que literalmente encerra a execução de uma <i>activity</i> , sendo feita após sua execução, remoção completa da aplicação da pilha e o encerramento completo do processo.

Tabla 1 – Métodos do ciclo de vida de uma aplicação

Em todo sistema operacional com suporte à memória virtual os processos são executados em diferentes regiões de memória, nenhum processo tem acesso direto à região de memória de outro processo ou *thread*. Dessa forma é necessário um mecanismo de comunicação entre processos, que é chamado de *binder* (GOMES et al., 2012).

2.2.3 Threads

O *Android* conta com um sistema sofisticado de gerenciamento de *threads*. Possui uma *thread* única nas aplicações padrão e *interface* com o usuário (HUBSCH, 2012). Quando uma aplicação é iniciada, o sistema cria uma *thread* para a aplicação chamada de *main* (primária) ou *UI () thread*. Esta *thread* é muito importante porque ela é encarregada dos eventos relacionados com a *interface* apropriada para o usuário. É também a *thread* que a aplicação interage com os componentes do *toolkit* do *Android* UI. Todos componentes que executam o mesmo processo

são instanciados pela UI *thread* e o despacho é feito pela mesma. Quando a aplicação executa trabalho intensivo em resposta a interação do usuário, o modelo *monothread* pode resultar em uma baixa performance. Quando a *thread* é bloqueada, nenhum evento é despachado, incluído eventos de desenho. Caso seja bloqueada por mais de 5 segundos, uma mensagem é mostrada para que o usuário possa fechar a aplicação. Existem duas regras para o modelo *monothread*: que não bloqueie a UI *thread* e não acesse o *toolkit* do *Android* UI fora da UI *thread* (MUNIZ, 2012).

2.3 Importância da Disponibilidade de Recursos Computacionais para *Android* na Execução de *Games*

Os jogos para dispositivos móveis evoluíram ao ponto de hoje apresentarem uma grande variedade de recursos, tais como gráficos em 3D. Desenvolver jogos que sejam capazes de apresentar esses recursos envolve fatores dependentes das características de *hardware* disponíveis a cada dispositivo, tais como capacidade de processamento e armazenamento. No entanto, os dispositivos móveis atuais apresentam entre si características muito particulares, possuindo diferente quantidade de memória disponível para os aplicativos desenvolvidos e capacidade de processamento (ALMEIDA et al., 2005).

2.3.1 *Cache no Android*

A memória *cache* oferece acesso rápido de alto custo. A *LRU (Least Recently Use) Cache* é usada para excluir o objeto utilizado menos recentemente antes da *cache* exceder o seu tamanho designado. Para escolher um tamanho adequado para um *LRUCache*, alguns fatores são considerados, tais como: a quantidade de imagens que estarão na tela ao mesmo tempo, quantas imagens precisam estar disponíveis pronto para entrar na tela, tamanho da tela, quais as dimensões e configuração dos *bitmaps*, frequência em que as imagens podem ser acessados. Não existe um tamanho de *cache* ou fórmula específica que se adapte a todas as aplicações. Um *cache* que é muito pequena causa sobrecarga adicional sem nenhum benefício, um *cache* que é muito grande pode causar exceções e deixar o resto do aplicativo com pouca memória para trabalhar (ANDROID, 2012).

2.3.2 *Tamanho de Tela*

A utilização da tecnologia de tela *touchscreen* trouxe uma nova gama de possibilidades. Telas maiores (com teclados virtuais) no lugar de teclados *qwerty*, oferecem melhores resoluções para vídeos e jogos (HAMANN, 2011). Como o sistema operacional *Android* é executado

em uma variedade de dispositivos que oferecem diferentes tamanhos de tela e densidades. O próprio sistema realiza o redimensionamento da aplicação, permitindo que o aplicativo funcione em diferentes tamanhos de tela e densidade. O *Android* trabalha com a unidade de medida dp (Density-Independent Pixel) para definir tamanhos dos objetos na tela (ANDROID, 2012).

O que define a qualidade de uma tela é a quantidade de *pixels* que a mesma possui. A principal tecnologia para tela de *smartphones* com *Android* é a tela Super AMOLED (*Active-Matrix Organic Light-Emitting Diode*), onde as imagens são muito nítidas e os *pixels* são imperceptíveis aos olhos humano. Uma tela AMOLED possui quatro camadas: uma de cátodo, uma orgânica, outra de circuitos TFT (Thin Film Transistor) e uma última com substratos. Esse conjunto permite que esse tipo de *display* transmita essas ordens para os *pixels* três vezes mais rápido do que as telas OLED (*Organic Light-Emitting Diode*) comuns, o que permite ver aplicações com maior nitidez. A tecnologia Super AMOLED são muito utilizadas no mercado pelos *smartphones Galaxys*, da *Samsung* (CARDOSO, 2013).

2.3.3 Bibliotecas

O *Android* possui um conjunto de bibliotecas, disponíveis para a criação de seus aplicativos como a *System C Library*, *Media Libraries*, *Surface Manager*, *Lib Webcore*, *SGL*, *3D libraries*, *Freetype* e *SQLite*. Tais bibliotecas permitem a manipulação de vídeos, imagens, sons, animações, banco de dados, etc (SILVA, 2013).

- ***Media Libraries***

A *media libraries* é baseada em *OpenCORE*, que garante a reprodução e gravação dos formatos populares de áudio e vídeo, e arquivos de imagens estáticas. Incluem os formatos MPEG4, H.264, MP3, AAC, AMR, JPG e PNG. Esta biblioteca é desenvolvida pela *PacketVideo's*,

- ***Surface Manager***

Biblioteca que gerencia o acesso ao subsistema de *display* do dispositivo, é capaz de compor gráficos em 2D e 3D a partir de aplicações de camada múltiplas.

- ***Bibliotecas 3D***

Baseada em *OpenGL ES 1.0* as bibliotecas usam, tanto aceleração por *hardware* na medida do possível, quanto renderização 3D por *software*, sendo neste caso altamente otimizado.

- ***FreeType***

Gerenciador de fontes, que facilita a renderização de fontes *TrueType*, é de código aberto

e otimizado para diversos tipos de plataformas, principalmente nos casos sistemas incorporado ou portáteis.

3 Descrição de Ferramentas de Análise e Teste de Desempenho, Dispositivos e Games

Para realização dos testes foram utilizadas ferramentas já existentes no mercado, ferramentas que permitem verificar o comportamento de recursos computacionais em dispositivos móveis na execução de aplicações. Neste trabalho foram realizados testes apenas na execução de *games*, pelo fato de serem aplicações que requerem um alto poder de processamento, e possuem um número crescente de usuários.

Neste capítulo são descritas as ferramentas de análise e teste de desempenho que serão utilizadas, para obter informações sobre o comportamento dos recursos computacionais na execução de *games*, nos dispositivos com sistema operacional *Android*, também são descritos os dispositivos e jogos que participaram dos testes.

3.1 DDMS (*Dalvik Debug Monitor Server*)

Dalvik Debug Monitor Server (DDMS), ou Servidor de Monitoramento de *Debug* do *Dalvik*, é um programa disponível no *Android SDK* (*Software Development Kit*), que permite o monitoramento de processos em execução no sistema, captura de *screenshots*, coleta de informações sobre *threads* e pilhas, iniciação de chamadas telefônicas, e envio de mensagens SMS de entrada. Dentre as suas funcionalidades encontra-se o tracing de chamadas de métodos, a qual é útil para análise dinâmica de aplicações. O DDMS pode ser executado em dispositivos emulados e em dispositivos reais, com a depuração via USB habilitada. Através da ferramenta *dmtracedump*, pode – se obter a representação em imagem do grafo das chamadas de métodos do arquivo gerado pelo DDMS como ilustrado na Figura 2, facilitando a visualização (BRAGA et al., 2013).

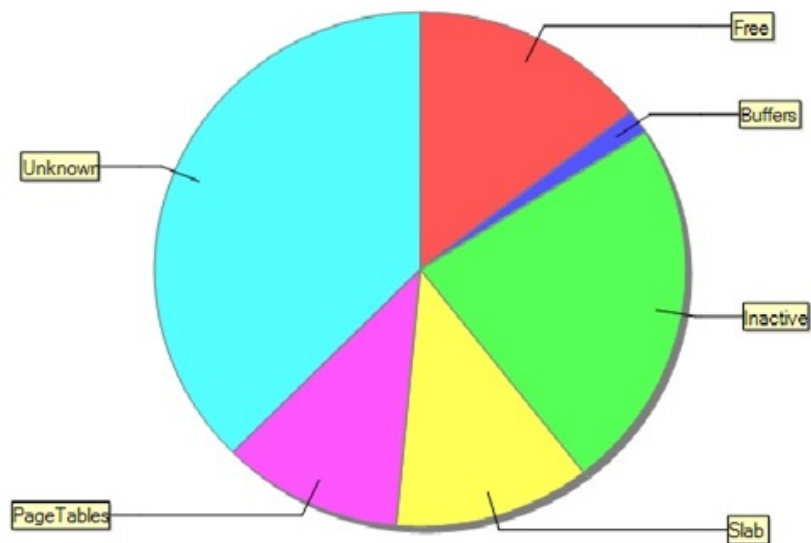


Figura 2 – DDMS- Uso de Memória

O gráfico ilustrado na Figura 1, mostra o percentual de uso da memória ou processador utilizados pelos recursos tais como quantidade de memória desconhecida (*Unknown*), área livre (*Free*), *buffers*, quantidade de memória inativa (*Inactive*), placa (*Slab*), tabela de páginas (*PageTables*), informações sobre processos, dentre outras. O percentual da utilização de cada recurso é exibido na tela no momento em que o mouse passa em cima de cada parte do gráfico. O *slab* é um local que guarda objetos e é composto por uma ou mais *frames* de páginas, uma *cache* refere-se ao número de *slabs* (partes) (UFAM, 2013).

Quando DDMS é iniciado, ele se conecta ao *Android Debug Bridge* (ADB). No momento que um dispositivo for conectado ao computador, um serviço de monitoramento de VM é criado entre ADB e o DDMS, que notifica o DDMS quando uma VM no dispositivo é iniciada ou encerrada. Isso acontece porque no *Android*, cada processo de um aplicativo é executado em sua própria máquina virtual. Cada VM expõe uma única porta que um depurador pode anexar. Uma vez que o serviço de monitoramento está executando, o DDMS recupera o código de identificação do processo da VM, via ADB, e abre uma conexão com depurador da VM, através do *daemon* do ADBD do dispositivo (FREIRE, 2013).

3.2 QtADB

O QtADB é desenvolvido em Qt (*framework* multiplataforma para desenvolvimento em C++) e utiliza o ADB para a comunicação com o aparelho. Qtadb é uma ferramenta com *interface* gráfica baseada no ADB, que permite o gerenciamento de aplicações para usuários *Android*. O programa é uma alternativa para o uso do ADB, pelo fato de possuir uma *interface* gráfica

que facilita o uso do mesmo pelos usuários. O Qtadb oferece suporte às principais funções do ADB, como gerenciador de arquivos, *backups*, restauração de cópia de segurança, captura de imagem de tela, gerenciador de aplicações, *interface Shell*, entre outras (MOZARIK, 2013). Na Figura 3, observa-se que cada ícone é uma saída que o programa QtADB disponibiliza. No ícone informação do telefone, são exibidas informações sobre nível da bateria (*Battery Level*) e de disponibilidade, uso e tamanho dos seguintes recursos do dispositivo: memória (*Data available*), sistema (*System available*), cartão SD (*Secure Digital Card*) (*sdcard available*) e sistemas de arquivo (*ext available*).

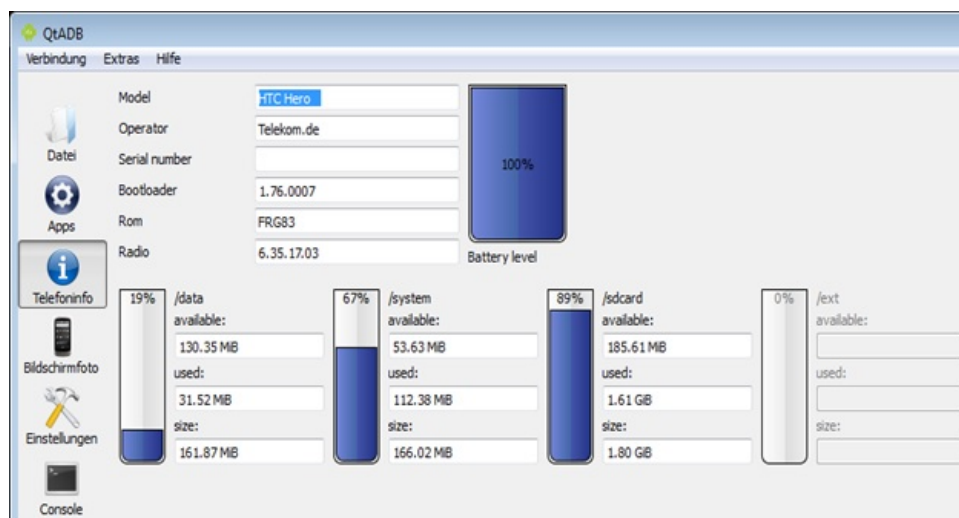


Figura 3 – QtADB

3.3 *Quadrant Standard*

É um aplicativo para *Android* que através da sua execução no dispositivo, pode-se obter informações que as outras ferramentas citadas disponibilizam, além de mostrar o processamento gráfico 2D e 3D e I/O de dados (GOOGLE, 2012). O aplicativo fornece um gráfico com uma comparação de desempenho entre o dispositivo em que está sendo executado e outros dispositivos com características diferentes, como ilustrado na Figura 4.

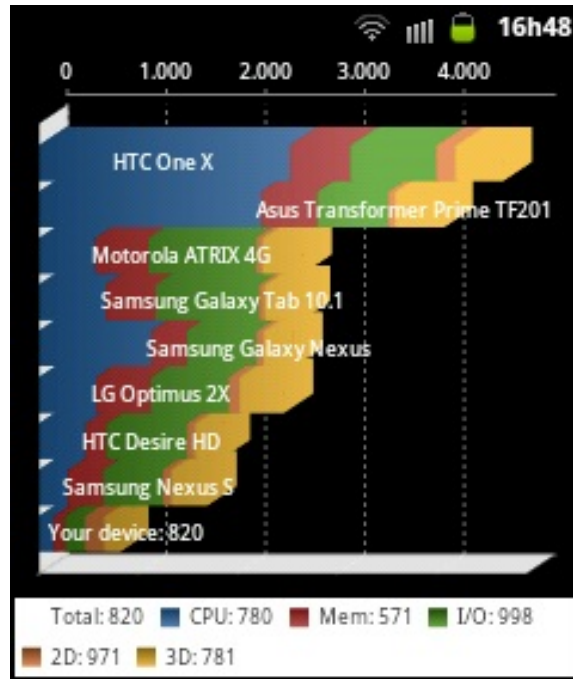


Figura 4 – Quadrant Standard

Como resultado da análise, o *Quadrant Standard* exibe um gráfico como o mostrado na Figura 4. O gráfico exibe informações de desempenho de memória, CPU, I/O e gráficos 2D e 3D do aparelho que está executando o aplicativo, e compara com informações de outros dispositivos de configurações diferentes. Cada barra do gráfico representa um dispositivo móvel como: *Samsung Galaxy Nexus*, *Motorola Atrix 4G*, *LG Optimus 2x*, dentre outros. A barra que contém *Your device* representa o dispositivo que está executando o aplicativo e na parte inferior é mostrada a quantidade detalhada do consumo dos recursos computacionais (memória, CPU, I/O, gráficos 2D e 3D) que são avaliadas no dispositivo.

A ferramenta QtADB apesar de mostrar informações relevantes sobre os recursos dos dispositivos, foi desenvolvida recentemente e possui algumas falhas e *bugs*, além de restrições quanto ao modelo do aparelho, essas falhas comprometem até mesmo o uso da ferramenta (QTADB, 2013). Por este motivo a ferramenta QtADB não será utilizada.

3.4 Especificações dos Dispositivos Utilizados na Realização dos Testes

O sistema *Android* possui um número grande de fatores que podem influenciar na sua performance, como a arquitetura, frequência do *clock* e número de núcleos de um processador, barramento e quantidade de memória, versão do *Android*, tamanho dos discos interno e externo, resolução da tela, entre vários outros (FREIRE, 2013). A Tabela 2 mostra os dispositivos utilizados na execução dos testes e suas especificações técnicas.

Nome e modelo do Dispositivo	Memória RAM	Processador	Versão	Tela	Resolução
_____ Samsung Galaxy Y-GT-S5360B	290 MB	832 MHz	2.3.6	3"	240x320
_____ Samsung Ace GT S5830c	278MB	832 MHz	2.3.6	3.5"	480x320
_____ Samsung Galaxy Pocket-GT-S5312B	512 MB	850 MHz	4.1.2	3"	240x320
_____ Samsung Galaxy S Duos-GT-S7562L	768 MB	1GHz	4.0	4"	800x480
_____ Sony Xperia E C1604	512 MB	1 GHz	4.1.1	3,5"	320x480
_____ Motorola RAZR i XT890	1GB	2GHz	4.0	4,3"	540x960
_____ LG Optimus L5 E615S	512MB	800MHz	4.0	4"	320x480

Tabela 2 – Dispositivos Utilizados na Execução dos Testes

Os principais parâmetros para escolha dos dispositivos a serem testados, foram a quantidade de memória RAM disponível no aparelho juntamente com a capacidade do processador, foram escolhidos dispositivos com pouca memória RAM (290 MB) até dispositivos com maior quantidade (1 GB). Quanto a capacidade de processamento foram escolhidos aparelhos que possuem um baixo poder de processamento (832 MHz) até aparelhos com poder de processamento mais elevado (2GHz). A utilização de dispositivos com diferentes configurações de memória e processador é importante pois oferecem uma variação no desempenho do jogos.

3.5 Descrição dos Jogos

Um dos principais motivos para seleção dos jogos selecionados para realização dos testes é o quesito popularidade ente usuários de *Android*. O site MOBILEGAMER (2013) divulgou uma lista com os 13 jogos mais populares do *Android* desde sua criação, e nessa lista citou os *games*, *Angry Birds Rio* e *Jetpack Joyride*. Uma outra pesquisa que mostra os 10 jogos mais populares no *Android* em 2013, inclui em sua relação os jogos (EM10TAQUE, 2013): *Fruit Ninja*, *Hill Climb Racing*, *Jetpack Joyride* e *Real Racing 3.O*. O *game Real Racing 3* é um jogo de corrida de automóveis, o mesmo foi substituído por um *game* do mesmo gênero: *Speed Car*. O motivo da substituição foi a falta de compatibilidade com algumas versões de *Android* de alguns dispositivos utilizados nos testes. Assim como outros *games* que foram citados pelos

sites não foram escolhidos, pelo mesmo motivo.

3.5.1 *Angry Birds Rio*



Figura 5 – Angry Birds Rio

Angry Birds Rio é um jogo desenvolvido pela Rovio Entertainment, onde o jogador irá ajudar a libertar os pássaros que estão presos na gaiola, eles foram aprisionados por um cara malvado e deixados a sua própria sorte na cidade do Rio de Janeiro, a tela inicial do *game* é demonstrada na Figura 5. O Jogador irá lançar os pássaros que não foram capturados em direção a pilha de gaiolas para libertar os demais. Uma boa mira e precisão serão necessárias para o sucesso desta operação. O jogador dispõe de uma fisga que permite lançá-los diretamente sobre as gaiolas, ou sobre estruturas, que ao serem destruídas geram pontos ao jogador. Existem vários tipos de pássaros. Alguns possuem vantagem para destruir determinado elemento, outros possuem características únicas e uma habilidade, que é ativada tocando na tela enquanto o pássaro voa. *Angry Birds* apresenta desafio e jogabilidade, além de horas de repetição com base na física. Cada nível requer lógica, habilidade e força para resolver, o tamanho do *game* equivale a 31 MB e a versão atual é a 1.8.0 (ROVIO, 2013).

3.5.2 *SpeedCar*

SpeedCar como demonstrado na Figura 6, é um jogo de corrida 3D que permite ao jogador obter uma sensação realista do efeito físico do carro. O jogador pode acelerar o carro, aumentando assim a velocidade. Para controlar a direção do carro é necessário inclinar o *smartphone* para direita ou esquerda. O objetivo é acelerar e desviar do maior número de obstáculos que conseguir em uma rua cheia de carros. Além disso, cones se espalham pelas ruas e as barreiras laterais também representam perigo, é um *game* desenvolvido pelo OoO studio e possui 3,6 MB de tamanho (GOOGLE, 2012).



Figura 6 – SpeedCar

3.5.3 Hill Climb Racing



Figura 7 – Hill Climb Racing

O *Hill Climb Racing* é um *game* em 2D desenvolvido pela *Fingersoft*, que possui como meta conduzir veículos, a qualidade do veículo aumenta à medida em que se joga. O Jogador passar por diferentes mapas e coleta moedas durante o percurso através de performances acrobáticas ou completando níveis. O primeiro veículo com que o jogador se deparará é um pequeno *jipe*. Portanto, o jogador pode adquirir um novo veículo usando as moedas coletadas. A Figura 7 mostra a tela inicial do *game*, o jogo possui 14 etapas em locais diferentes como: Campo, Deserto, Ártico e Lua. O *Hill Climb Racing* possui ícones autoexplicativos presentes nos menus e os comandos funcionam através dos pedais de freio e de aceleração que são de fáceis manuseio, o tamanho do *game* é 16 MB (GOOGLE, 2012).

3.5.4 *Jetpack Joyride*



Figura 8 – Tela Inicial do Jetpack Joyride

O *Jetpack Joyride* é um game de ação inspirado no *Fruit Ninja* e desenvolvido pela *Halfbrick*, onde o herói *Barry Steakfries* encontra um laboratório secreto e deve comandar os *jetpacks* experimentais que estão nas garras de cientistas malfeitores. Após a decolagem o herói pode subir e descer, chovendo balas, bolhas, arco-íris e *lasers* para baixo. Ao longo de cada jogo moedas vão sendo coletadas e podem ser utilizadas para comprar novos equipamentos em *The Stash*. O jogador escolhe o *jetpack* favorito, roupa *snazzy* e demais de itens, quanto ao tamanho ele varia de acordo com o dispositivo, a Figura 8 exibe a tela inicial do *Jetpack Joyride* (GOOGLE, 2012).

3.5.5 *Fruit Ninja*



Figura 9 – Game Fruit Ninja

É um jogo de habilidade que tem como desenvolvedor a *Halfbrick*, onde o jogador deve cortar as frutas que estão sendo lançadas no ar como se fosse um ninja. Porém deve ficar atento para não explodir as bombas que estão misturadas entre as frutas lançadas. A figura 9 ilustra o cenário do game, o jogo disponibiliza vários tipos de lâminas e vários planos de fundo, permite

ainda que o jogador obtenha conhecimento sobre as frutas sábio ninja *Sensei*, já o tamanho do *game* varia de acordo com o dispositivo, a tela inicial do *Fruit Ninja* esta demonstrada na Figura 9 (GOOGLE, 2012).

4 Resultados e Discussões

Com a ferramenta DDMS pode-se obter informações sobre o monitoramento de processos em execução no sistema, captura de *screenshots*, coleta de informações sobre *threads* e pilhas, iniciação de chamadas telefônicas, e envio de mensagens SMS de entrada. Dentre as suas funcionalidades encontra-se o *tracing* de chamadas de métodos, a qual é útil para análise dinâmica de aplicações. Através da ferramenta *dmtracedump*, pode – se obter informações sobre uso da memória ou processador utilizados pelos recursos, tais como quantidade de memória desconhecida (*Unknown*), área livre (*Free*), *buffers*, quantidade de memória inativa (*Inactive*), placa (*Slab*), tabela de páginas (*PageTables*) e informações sobre processos (BRAGA et al., 2013).

O aplicativo *Quadrant Standard* permite acesso a informações sobre o uso de memória, CPU, processamento gráfico 2D e 3D e I/O de dados. (GOOGLE, 2012).

Neste capítulo serão descritos todos os resultados obtidos na realização dos testes por meio do uso das ferramentas: aplicativo *Quadrant Standard* e da ferramenta DDMS.

4.1 Testes de Execução

Os jogos que serão utilizados na realização dos testes são os seguintes: *Angry Birds*, *Speed Car*, *Hill Climb Racing*, *Jetpack Joyride*, *Fruit Ninja*. Cada *game* será executado individualmente em cada dispositivo e analisado individualmente em cada uma das duas ferramentas que serão utilizada. Na execução dos testes nas ferramentas DDMS e *Quadrant Standard* todos os *games* estavam na primeira fase, sendo executado no tempo de cerca de 1 minuto.

4.1.1 Execução com Ferramenta DDMS

Em todas as tabelas de testes de execução da ferramenta DDMS, contém as respectivas colunas: Uso de CPU e dados de memória: Inativa, *Slab*, *Page*, *Unknow*, *Free*, *Buffer*.

- A coluna Uso da CPU, está relacionado com a quantidade de tempo de processamento que a aplicação utiliza no momento de sua execução.
- A coluna Inativa, faz referência a parte da memória que o sistema já endereçou, mas deixa o registro guardado caso haja necessidade de recuperar os valores.

- *Slab*, a área da memória principal que contém uma *cache* é dividida em *slabs*, cada *slab* consiste de um ou mais *frames* de páginas contíguos que contém os objetos alocados e livres (UFAM, 2013).
- *PageTables* (Tabela de Páginas), a memória é dividida em pedaços de tamanho fixo (páginas), e segmentos de código são alocados nestes e mapeados utilizando-se uma tabela de páginas, ao invés de alocação de todo código de uma única vez (GOMES et al., 2012).
- *Unknow*, representa a parte do gerenciamento de memória que ficou desconhecido.
- *Free*, essa é a quantidade de RAM que não está sendo usada.
- *Buffer*, é um repositório que guarda os últimos endereços acessados, para que não haja necessidade de fazer múltiplos acessos as páginas. Antes de realizar a procura do endereço utilizando as tabelas de página o sistema busca o endereço no *Buffer* (GOMES et al., 2012).

4.1.2 Execução com Ferramenta *Quadrant Standard*

No aplicativo *Quadrant* os resultados das análises são mostradas por meio de uma pontuação, quanto maior a pontuação de um dispositivo em um determinado quesito, melhor foi o seu desempenho. O aplicativo exibe o desempenho de memória, CPU, entrada e saída de dados I/O, recursos gráficos 2D e 3D e por fim exibe uma avaliação total sobre o dispositivo. Nos testes com o aplicativo *Quadrant Standard*, o jogo fica executando em segundo plano enquanto o aplicativo em primeiro plano, porém o *game* continua sendo executado em um processo.

4.2 *Game Angry Birds*

4.2.1 Execução com Ferramenta DDMS

Nome do Dispositivo	CPU	Memória					
	Uso	Inativa	<i>Slab</i>	<i>Page</i>	<i>Unknow</i>	<i>Free</i>	<i>Buffer</i>
<i>Galaxy Y</i>	75,77%	51%	19%	18%	4%	5%	2%
<i>Galaxy Ace</i>	64,68%	35%	25%	20%	13%	7%	0%
<i>Galaxy Pocket</i>	32,39%	12%	11%	5%	67%	5%	0%
<i>Galaxy S Duos</i>	46,63%	19%	9%	7%	48%	15%	3%
<i>Sony Xperia</i>	60,67%	13%	6%	3%	74%	3%	1%
<i>Motorola RAZR i</i>	65,47%	12%	2%	1%	81%	3%	1%
<i>LG L5</i>	52,58%	9%	4%	2%	79%	4%	2%

Tabela 3 – Ferramenta – DDMS game Angry Birds

Com a ferramenta DDMS o **dispositivo *Galaxy Pocket*** que possui um processador com velocidade de 850 MHz como mostrado na Tabela 2, apresentou o menor consumo de tempo de processamento na execução do *Angry Birds*, equivalente a **32,39%** como apresentado na Tabela 3. O dispositivo que alcançou o maior tempo de processamento foi o *Galaxy Y* como cerca de 75,77%. O processamento do *game Angry Birds* apresentou um uso de CPU com uma média aproximada em 56,88% entre todos os dispositivos.

Quanto maior a quantidade de memória inativa, mais rapidamente um aplicativo que já esteve em execução anteriormente poderá ser carregado na memória. **Os dispositivos que tingiram as maiores taxas de memória inativa foram: *Galaxy Y* com 51% equivalente a aproximadamente 147MB e o *Galaxy S Duos* com 19% que é equivalente a aproximadamente 145MB.** Na coluna taxa de *slabs*, quanto maior for seu valor, então maior será o espaço disponível para alocação de objetos na memória, nesse quesito o *dispositivo Motorola RAZR i* atingiu a menor taxa, apenas 2% destinados para *slab*.

Em relação a coluna *PageTables*, quanto maior for sua taxa, significa mais espaços disponíveis na memória para novos processos, proporcionando assim melhor desempenho, os dispositivos *Galaxy Ace* e *Galaxy S Duos* apresentaram as melhores taxas se levado em conta a quantidade de memória do dispositivo, *Galaxy Ace* (55,6 MB) e *Galaxy S Duos* (53,76).

As maiores taxas de uso de memória desconhecida ficaram por conta dos dispositivos *LG L5* (79%) e *Motorola RAZR i* (81%), como o uso é “desconhecido” esse espaço da memória pode armazenar parte do *game* ou não. O dispositivo *Galaxy S Duos* apresentou a melhor taxa de memória *free* (15%), indicando uma melhor disponibilidade de memória livre. Quanto aos valores relacionados ao uso de *buffer*, todos os dispositivos apresentaram uma taxa menor que 5%, a utilização do *buffer* permite um melhor desempenho no momento de busca de dados, sendo que dois dispositivos *Galaxy Ace* e *Galaxy Pocket* apresentaram taxas iguais a 0%.

4.2.2 Ferramenta *Quadrant Standard*

Dispositivo	Memória	CPU	I/O	2D	3D	Total
<i>Galaxy Y</i>	586	775	926	980	799	813
<i>Galaxy Ace</i>	629	776	926	327	578	647
<i>Galaxy Pocket</i>	1534	1841	3753	971	1704	1961
<i>Galaxy S Duos</i>	1604	2553	4958	314	1504	2187
<i>Sony Xperia</i>	1726	2661	4490	551	1544	2194
<i>Motorola RAZR i</i>	4609	5646	4113	501	1929	3360
<i>LG L5</i>	1351	1953	3482	629	1258	1735

Tabela 4 – Ferramenta – *Quadrant Standard* game *Angry birds*

Como demonstrado na Tabela 4 o *game Angry Birds* apresentou um melhor desempe-

no dispositivo *Motorola RAZR i* levando em consideração (Memória, CPU, I/O, 2D, 3D), obtendo um resultado de 3360. Porém não foi o melhor em todos os quesitos, pois no quesito I/O os dispositivos *Galaxy S Duos*(4958) e *Xperia* (4490) apresentaram um melhor desempenho. Na coluna 2D vários dispositivos apresentaram melhores resultados que o *Motorola RAZR i*, como o *Galaxy Pocket* com 971 pontos.

4.3 Testes na execução do game *Speed car*

4.3.1 Execução com Ferramenta DDMS

Nome do Dispositivo	CPU	Memória					
	Uso	Inativa	Slab	Page	Unknow	Free	Buffer
<i>Galaxy Y</i>	58,59%	48%	24%	20%	3%	5%	1%
<i>Galaxy Ace</i>	48,47%	44%	20%	18%	0%	16%	1%
<i>Galaxy Pocket</i>	30,46%	11%	10%	5%	41%	33%	0%
<i>Galaxy S Duos</i>	26,31%	20%	8%	7%	39%	25%	1%
<i>Sony Xperia</i>	32,36%	15%	6%	3%	63%	12%	1%
<i>Motorola RAZR i</i>	35,29%	12%	2%	1%	78%	6%	1%
<i>LG L5</i>	42,49%	9%	5%	2%	76%	4%	4%

Tabela 5 – Ferramenta – DDMS game Speed car

Na execução do *Speed Car*, em relação ao consumo de tempo de processamento o *game* apresentou uma média de aproximadamente 39% entre todos os dispositivos testados. Sendo que o *Galaxy S Duos* apresentou a menor taxa de tempo (26,31%), em relação a sua própria capacidade de processamento.

Quanto ao gerenciamento de memória o *Speed car* apresentou na coluna referente a quantidade de memória inativa valores com grande variação. O *Motorola RAZR i* demonstrou uma taxa pequena de memória inativa (12%) como visto na Tabela 5, em relação aos demais dispositivos. Porém quando se leva em consideração seu tamanho de memória, percebe-se que o mesmo reservou aproximadamente 122 MB para memória inativa, assim como o *Galaxy Ace* que também disponibilizou aproximadamente 122 MB. O *Galaxy Y* disponibilizou cerca de 139 MB para memória inativa. Portanto os dispositivos que reservaram maiores espaços para memória inativa foram *Galaxy y*, *Galaxy Ace* e *Motorola RAZR i*.

O dispositivo *Motorola RAZR i* apresentou a menor taxa referente a coluna *slab* 2%, quanto menor for o valor de *slab*, então menor o espaço disponível para alocação de objetos na memória. No quesito *pageTable* o *Motorola RAZR i* também atingiu a menor porcentagem cerca de 1%, ou seja, menos espaços disponíveis na memória para novos processos. Já em relação a coluna *Unknow* (uso desconhecido, podendo conter partes do *game* ou não) o dispositivo *Motorola RAZR i* apresentou a maior taxa dentre os demais dispositivos 78% cerca de aproximadamente 798 MB. O dispositivo *Sony Xperia* apresentou um a melhor taxa de

memória livre (*free*), o dispositivo ficou com 192 MB livre. No quesito *Buffer* (quanto menor a quantidade de *buffer*, maior a necessidade de fazer múltiplos acessos as páginas de memória) o *Lg L5* apresentou a maior taxa 4%, que representa aproximadamente 20 MB de memória do dispositivo.

4.3.2 Ferramenta *Quadrant Standard*

Dispositivo	Memória	CPU	I/O	2D	3D	Total
<i>Galaxy Y</i>	558	714	1020	786	629	1
<i>Galaxy Ace</i>	936	723	1123	300	683	753
<i>Galaxy Pocket</i>	1260	1682	3496	971	1476	1777
<i>Galaxy S Duos</i>	1631	2627	4902	280	1447	2177
<i>Sony Xperia</i>	1658	2653	4395	542	1567	2163
<i>Motorola RAZR i</i>	6950	5504	4184	521	1974	3827
<i>LG L5</i>	1228	1795	3177	565	1245	1602

Tabela 6 – Ferramenta – *Quadrant Standard* game *Speed Car*

Na execução do *Speed Car* o dispositivo *Galaxy Y* apresentou a menor pontuação, 741 no Total demonstrado na Tabela 6. Já entre os dispositivos que possuem o mesmo tamanho de memória 512 MB como, *Galaxy Pocket*, *Xperia*, *LG L5* o *Xperia* apresentou a melhor pontuação 1658. **O dispositivo *Motorola RAZR i* obteve a melhor pontuação(3827)** como demonstrado na Tabela 6, levando em consideração todos os recursos analisados.

4.4 Testes na execução do game *Hill Climb Racing*

4.4.1 Execução com Ferramenta DDMS

Nome do Dispositivo	CPU	Memória					
	Uso	Inativa	Slab	Page	Unknow	Free	Buffer
<i>Galaxy Y</i>	60,61%	40%	21%	20%	10%	8%	1%
<i>Galaxy Ace</i>	55,55%	54%	19%	18%	0%	7%	2%
<i>Galaxy Pocket</i>	29,32%	24%	11%	6%	47%	12%	1%
<i>Galaxy S Duos</i>	39,57%	22%	8%	7%	42%	18%	2%
<i>Sony Xperia</i>	33,43%	16%	6%	3%	64%	10%	1%
<i>Motorola RAZR i</i>	32,27%	13%	2%	1%	79%	3%	2%
<i>LG L5</i>	40,50%	10%	4%	2%	81%	2%	2%

Tabela 7 – Ferramenta – DDMS game *Hill Climb Racing*

Em relação ao consumo de tempo de processamento, na execução do *Hill Climb Racing* a maior parte dos dispositivos apresentaram uma média de aproximadamente 41% . O *Galaxy y* foi o dispositivo que indicou a maior taxa de consumo de tempo (60,61%) em relação ao seu

próprio contexto, como demonstrado na Tabela 7. O *Galaxy Pocket* obteve o menor tempo de processamento, apenas 29,32%.

Quanto ao gerenciamento de memória relacionado aos dados de memória inativa (quanto maior a quantidade de memória inativa, mais rapidamente um aplicativo que já esteve em execução anteriormente poderá ser carregado na memória), os dispositivos ***Galaxy Pocket*** e ***Lg 15*** apesar de possuírem a mesma quantidade de memória RAM, apresentaram números bem distintos *Galaxy Pocket* (22%, aproximadamente 122 MB) e *Lg 15* (10%, aproximadamente 51 MB), o *Galaxy Ace* apresentou a maior porcentagem cerca de 40%. Em relação as colunas *slab* (quanto menor for seu valor, então menor será o espaço disponível para alocação de objetos na memória) e *pagetable* (quanto menor for a porcentagem, significa menos espaços disponíveis na memória para novos processos) o ***Motorola RAZR i*** apresentou as menores taxas (2% e 1%). Já referente a coluna *Unknow* o *Galaxy Ace* apresentou 0%, ou seja, toda sua memória foi utilizada no processamento do *game* de forma que nenhuma parte ficou desconhecida. O dispositivo que obteve a melhor taxa de memória *free*, ou seja, obteve mais espaço de memória livre foi o ***Galaxy S Duos*** com 18%, a média de memória livre entre todos os aparelhos foi apenas de aproximadamente 8,5%. A taxa de *buffer* em todos os dispositivos variou entre 1% e 2%, ou seja, haverá a necessidade de fazer múltiplos acessos as páginas de memória.

4.4.2 Ferramenta – *Quadrant Standard Hill Climb Racing*

Dispositivo	Memória	CPU	I/O	2D	3D	Total
<i>Galaxy Y</i>	513	681	896	855	670	723
<i>Galaxy Ace</i>	870	746	1226	314	381	707
<i>Galaxy Pocket</i>	1450	1486	4556	1000	1765	2051
<i>Galaxy S Duos</i>	1561	2527	4347	303	1432	2034
<i>Sony Xperia</i>	1537	2387	4309	551	1192	1995
<i>Motorola RAZR i</i>	6869	5664	4100	505	2006	3829
<i>LG L5</i>	1392	1833	3716	595	1182	1744

Tabela 8 – Ferramenta – *Quadrant Standard game Hill Climb Racing*

Na execução do *Hill Climb Racing* o dispositivo que apresentou a menor pontuação no Total foi o *Galaxy Ace* com (707), demonstrado na Tabela 8, **novamente o dispositivo *Motorola RAZR i* ficou com a melhor pontuação no Total (3829)**. *Galaxy Pocket*, *Galaxy S Duos* e *Xperia* apresentaram pontuação maior que o *Motorola RAZR i* no quesito I/O, *Galaxy Pocket* (4556), *Galaxy S Duos* (4347) e *Xperia* (4309), enquanto o *Motorola RAZR i* apresentou 4100. Na coluna 2D vários dispositivos obtiveram pontuações melhores que o *Motorola RAZR i* (505), sendo que a melhor pontuação na coluna 2D ficou por conta do *Galaxy Pocket* com 1000 pontos.

4.5 Testes na execução do game *Jetpack Joyride*

4.5.1 Execução com Ferramenta DDMS

Nome do Dispositivo	CPU	Memória					
	Uso	Inativa	Slab	Page	Unknow	Free	Buffer
<i>Galaxy Y</i>	79,81%	37%	20%	18%	18%	5%	1%
<i>Galaxy Ace</i>	72,75%	36%	25%	19%	06%	12%	2%
<i>Galaxy Pocket</i>	39,44%	11%	11%	5%	61%	12%	0%
<i>Galaxy S Duos</i>	46,55%	15%	7%	6%	54%	17%	1%
<i>Sony Xperia</i>	42,49%	11%	5%	3%	77%	4%	0%
<i>Motorola RAZR i</i>	37,31%	10%	2%	1%	84%	3%	0%
<i>LG L5</i>	37,53%	9%	4%	2%	78%	6%	1%

Tabela 9 – Ferramenta – DDMS game *Jetpack Joyride*

Em relação ao consumo de tempo de processamento o game *Jetpack Joyride* apresentou uma média de 50,84% em relação a todos os dispositivos. Os dispositivos Lg L5 e *Galaxy Y* apesar de possuírem quase a mesma velocidade de processamento como observado na Tabela 2, *Galaxy Y* (832 MHz) e Lg L5(800 MHz), a taxa de uso do processador entre eles teve uma diferença de aproximadamente 42,28%. O Motorola *RAZR i* obteve a taxa mais baixa (37,31%) de uso da CPU.

O game *Jetpack Joyride* consumiu no quesito memória inativa valores que oscilaram entre 37% e 9% dependendo do dispositivo, no dispositivo **Lg L5** apresentou o menor valor, apenas 9% equivalente a aproximadamente 46 MB do seu tamanho de memória RAM disponível, quanto menor a quantidade de memória inativa, menor a quantidade de registro de aplicativos que estiveram em execução anteriormente na memória. Quanto aos valores destinados a *slab* (quanto menor for seu valor, então menor será o espaço disponível para alocação de objetos na memória) e *page* (quanto menor for a porcentagem, significa menos espaços disponíveis na memória para novos processos) os aparelhos *Motorola RAZR i* e LG L5 apresentaram as menores taxas, *Motorola RAZR i* (2% e 1%) e LG L5 (4% e 2%), apesar da diferença na porcentagem o tamanho de memória dedicado a *slab* e *page* foram os mesmos (*slab* 20,48 MB e *page* 10,24 MB).

O uso de memória desconhecido (*Unknow*) ficou por conta do *Motorola RAZR i* apresentando cerca de 84%. O game apresentou menor disponibilidade de memória livre (*free*) no dispositivo *Galaxy y* com valor de 5% equivalente apenas a 14,5 do tamanho total de memória disponível no dispositivo, já a maior disponibilidade de memória *free* ficou por conta do *Galaxy S Duos* com 17%. Os dispositivos *Galaxy Pocket*, *Sony Xperia* e *Motorola RAZR i* na execução do game *Jetpack Joyride* apresentaram taxa de *Buffer* igual a 0%, quanto menor a taxa de relacionada ao *buffer* maior a necessidade de fazer múltiplos acessos as páginas de memória.

4.5.2 Ferramenta *Quadrant Standard*

Dispositivo	Memória	CPU	I/O	2D	3D	Total
<i>Galaxy Y</i>	508	775	941	952	795	794
<i>Galaxy Ace</i>	511	746	1072	103	637	614
<i>Galaxy Pocket</i>	938	1873	4638	51	1670	1834
<i>Galaxy S Duos</i>	1685	2772	5093	330	1568	2290
<i>Sony Xperia</i>	1386	2389	4176	233	1055	1848
<i>Motorola RAZR i</i>	8495	5503	4074	490	2043	4121
<i>LG L5</i>	1375	1901	4093	595	894	1772

Tabela 10 – Ferramenta – *Quadrant Standard* game *Jetpack Joyride*

O *Jetpack Joyride* apresentou melhor desempenho quando executado no **Motorola RAZR i**, obtendo um Total de 4121, apresentados na Tabela 10, obteve as melhores pontuações no quesito memória (8495), CPU (5503) e 3D (2043), já nos quesitos I/O e 2D obteve uma pontuação menor que outros dispositivos, como *Galaxy S Duos* que no quesito I/O obteve o melhor pontuação cerca de 5093, na coluna referente aos recursos gráficos 2D o melhor dispositivo foi o Lg L5 com 595. O dispositivo que apresentou a pior pontuação na execução do *Jetpack Joyride* foi o *Galaxy Ace* atingindo no Total 614 pontos.

4.6 Testes na execução do game *Fruit Ninja*

4.6.1 Execução com Ferramenta DDMS

Nome do Dispositivo	CPU	Memória					
	Uso	Inativa	Slab	Page	Unknow	Free	Buffer
<i>Galaxy Y</i>	63,64%	37%	19%	16%	22%	5%	1%
<i>Galaxy Ace</i>	55,56%	34%	20%	17%	26%	6%	1%
<i>Galaxy Pocket</i>	40,40%	14%	13%	6%	65%	2%	1%
<i>Galaxy S Duos</i>	48,49%	25%	8%	7%	43%	17%	1%
<i>Sony Xperia</i>	47,51%	16%	6%	3%	71%	3%	1%
<i>Motorola RAZR i</i>	30,25%	13%	2%	1%	79%	4%	1%
<i>LG L5</i>	31,47%	9%	4%	2%	82%	1%	1%

Tabela 11 – Ferramenta – *DDMS* game *Fruit Ninja*

O game *Fruit Ninja* apresentou um menor consumo de tempo de processamento no dispositivo **Motorola RAZR i (30,25%)**. O game apresentou uma média de aproximadamente 45% de consumo tempo da CPU. O *Galaxy Y* obteve a maior taxa (63,64%).

No quesito memória inativa o game atingiu a menor taxa quando executado no dispositivo **Lg L5 utilizando apenas 9%** do tamanho total da memória RAM, obtendo assim a menor taxa de memória inativa entre os demais dispositivos, ou seja, menor a quantidade de registro de aplicativos que estiveram em execução anteriormente na memória. Na coluna *slab* o dispositivo *Galaxy Pocket* apresentou uma taxa de 13% que equivale a aproximadamente 66 MB

de memória RAM, ficando como o dispositivo que mais dedicou memória para esse quesito, quanto maior o valor de *slab*, maior será o espaço disponível para alocação de objetos na memória. O dispositivo *Motorola RAZR i* reservou a menor taxa, apenas 1% da sua memória para *pageTable*, ou seja, menos espaços disponíveis na memória para novos processos. O *Motorola RAZR i* atingiu a maior taxa para *Unknow* (uso desconhecido) cerca de 79% que é equivalente a aproximadamente 808 MB. O dispositivo **Lg I5 apresentou a menor taxa (1%)** de memória livre (*free*) na execução do *game*. Todos os dispositivos apresentaram na execução do *game Fruit Ninja* apenas 1% de memória dedicado ao *Buffer*, ou seja, maior a necessidade de fazer múltiplos acessos as páginas de memória.

4.6.2 Ferramenta *Quadrant Standard*

Dispositivo	Memória	CPU	I/O	2D	3D	Total
<i>Galaxy Y</i>	618	776	1185	971	784	867
<i>Galaxy Ace</i>	947	776	1052	333	726	767
<i>Galaxy Pocket</i>	1134	1364	4790	962	1244	1899
<i>Galaxy S Duos</i>	1718	2742	4762	317	1427	2193
<i>Sony Xperia</i>	1460	2410	4840	394	1400	2101
<i>Motorola RAZR i</i>	8727	5603	4131	533	1932	4185
<i>LG L5</i>	1334	1841	4071	570	1065	1776

Tabela 12 – Ferramenta – *Quadrant game Standart Fruit Ninja*

Na execução do *Fruit Ninja* o ***Motorola RAZR i* ficou com o Total de 4185**, obtendo assim o melhor desempenho. O dispositivo *Galaxy S Duos* apresentou a segunda melhor pontuação cerca de 2193 pontos. *Galaxy Ace* obteve menos pontos que os demais dispositivos nos quesitos memória, I/O (1052), 2D (333) e 3D (726), apresentando no Total cerca de 767 pontos.

4.7 Análise Geral da Ferramenta DDMS

No quesito tempo de CPU, na execução do *Angry Birds*, o *Galaxy Pockete* obteve a menor taxa. Enquanto na execução do *Speed Car* o dispositivo *Galax S Duos* atingiu o menor tempo de CPU. No *game Hill Cimb* os dispositivos que obtiveram as menores taxa de tempo foram *Galaxy Pocket* e *Motorola RAZR i*. O dispositivo *Motorola RAZR i* também atingiu a menor taxa de tempo de processamento na execução dos *games: Jetpack Joyride e Frit Ninja*. Quanto ao gerenciamento de memória o dispositivo *Galaxy S Duos* apresentou em todos os casos exceto na execução do *Speed car*, as maiores taxas destinada a memória livre. Porém com essa ferramenta não podemos definir qual *game* obteve melhor desempenho em um determinado aparelho, pelo fato dos *games* apresentarem uma grande variação nos resultados, ou seja, não houve um dispositivo que obteve as melhores taxas em todos os quesitos.

4.8 Análise Geral da Ferramenta *Quadrant Standard*

Na execução de todos os *games* que participaram da análise o dispositivo *Motorola RAZR i* com maior configuração entre os demais dispositivos (1 GB de memória RAM e 2GHz de velocidade de processamento) obteve as melhores pontuações, apresentando uma média de aproximadamente 3895 pontos no Total. Porém o *Motorola RAZR i* não obteve as melhores pontuações em todos os quesitos, nos quesitos I/O e 2D por vezes ele foi superado por outros dispositivos como o *Galaxy S Duos*. O dispositivo *Galaxy S Duos* atingiu a segunda melhor pontuação no Total, atingiu uma média de aproximadamente 2181 pontos. Os *games* apresentaram um desempenho menos favorável no dispositivo *Galaxy Ace*, apresentando uma média de aproximadamente 696 pontos no Total. vel no dispositivo Galaxy Ace, apresentando uma média aproximadamente 696 pontos no Total .

Todos os jogos executaram em todos os aparelhos, apenas o *game Jetpack Joyride* nos dispositivos *Galaxy y* e *Galax Ace*, apresentou uma alteração em seus gráficos, onde a imagem do personagem não aparecia, mas sim uma faixa branca.

5 Conclusão

Na realização dos testes, tanto o aplicativo *Quadrant Standart* quanto a ferramenta DDMS atingiram os resultados esperados, mostrando informações detalhadas sobre o uso dos recursos computacionais dos dispositivos. As duas ferramentas permitem que desenvolvedores obtenham informações importantes sobre a execução dos *gamers*, como consumo de memória, CPU, detalhes sobre gráficos 2D e 3D e entrada e saída de dados, podendo utilizar tais informações para otimização do processador, melhor gerenciamento de espaços de memória, aperfeiçoamento na utilização das bibliotecas que se referem aos gráficos, entre outros.

Com a ferramenta *Quadrant Standart* todos os *gamers* que participaram da análise o dispositivo *Motorola RAZR i* obteve as melhores pontuações, apresentando uma média de aproximadamente 3895 pontos no Total. Os *gamers* apresentaram um desempenho menos favorável no dispositivo *Galaxy Ace*, apresentando uma média de aproximadamente 696 pontos no Total. Com a ferramenta DDMS não podemos definir qual *game* obteve melhor desempenho em um determinado aparelho, pelo fato dos games apresentarem uma grande variação nos resultados, ou seja, não houve um dispositivo que obteve as melhores taxas em todos os quesitos.

Com os valores disponibilizados os desenvolvedores podem melhor gerenciar de acordo com a porcentagem ou pontuação atingida em cada recurso, e realizarem avaliações referente a performance de um *game* em um determinado dispositivo. Com os resultados dos testes, os usuários poderão utilizar essas informações e escolher o dispositivo que melhor atende suas reais necessidades.

Referências

- ABINEE. **Vendas de Smartphones puxam mercado de celulares.** 2013. Disponível em: <<http://www.abinee.org.br/noticias/com235.htm>>. Acesso em: 20 nov. 2013.
- ALMEIDA, H. O.; FILHO, E. C. L.; NOGUEIRA, W. F. **Plataformas para Desenvolvimento de Jogos para Celulares.** 2005. Disponível em: <<http://www.dcc.ufla.br/infocomp/artigos/v4.1/art07.pdf>>. Acesso em: 02 agos. de 2013.
- ANDROID. **Android Developers.** 2012. Disponível em: <<http://developer.android.com/guide/index.html>>. Acesso em: 20 Junho 2013.
- AQUINO, J. S. F. **Plataforma de Desenvolvimento para Dispositivos Móveis.** 2007. Disponível em: <<http://www-di.inf.pucrio.br/~endler/courses/Mobile/Monografias/07/Android-Juliana-Mono.pdf>>. Acesso em: 04 out. de 2013.
- BESSA, A. **Computadores e Sistemas.** 2013. Disponível em: <http://belojardim.ifpe.edu.br/apostilas/BOU_Computadores_e_sistemas.pdf>. Acesso em: 06 dez. de 2014.
- BORDIN, M. V. **Introdução a Arquitetura Android.** 2012. Disponível em: <<sites.setrem.com.br/stin/2012/anais/Maycon.pdf>>. Acesso em: 07 julho de 2013.
- BRAGA, Alexandre Melo et al. **Introdução à Segurança de Dispositivos Móveis Modernos – Um Estudo de Caso em Android.** 2013. Disponível em: <<http://dainf.ct.utfpr.edu.br/~maziero/lib/exe/fetch.php/ceseg:2012-sbseg-mc2.pdf>>. Acesso em: 20 out. de 2013.
- CARDOSO, P. **O que é tela Super AMOLED e como funciona?** 2013. Disponível em: <<http://www.techtudo.com.br/artigos/noticia/2011/06/como-funciona-uma-tela-super-amoled.html>>. Acesso em: 20 jan. de 2014.
- CARISSIMI, A. S; OLIVEIRA, R. S.; TOSCANI, S. S. **Sistemas Operacionais.** 2011. Disponível em: <<http://www.lume.ufrgs.br/bitstream/handle/10183/19242/000102159.pdf>>. Acesso em: 06 dez. de 2013.
- CASTRO, R.O.; JÚNIOR, M. A. P. **Um estudo de caso da plataforma Android com Interfaces Adaptativas.** 2011. Disponível em: <http://www.fgh.escoladenegocios.info/revistaalumni/artigos/Artigo_Marco%20Antonio.pdf>. Acesso em: 13 ago. de 2013.
- COSTA, C. **Projetando Controladores Digitais com FPGA.** 2006. Disponível em: <<http://s3.amazonaws.com/ppt-download/temp-130527200307phpapp01.pdf?responsecontentdisposition=attachment&Signature=%2B6SVPu%>>

2FV4r52JIquelMabI46d%2BI%3D&Expires=1373745857&AWSAccessKeyId=AKIAIW74DRRRQSO4NIKA>. Acesso em: 09 julho de 2013.

EM10TAQUE. **10 jogos mais populares no *Android* em 2013**. 2013. Disponível em: <<http://www.em10taque.com/help10k/10-jogos-populares-android-2013/>>. Acesso em: 15 jan. de 2014.

FREIRE, J. G. F. **Análise de Desempenho de Plataformas para Desenvolvimento com o Sistema Operacional *Android* (*Android Debug Bridge*)**. 2013. Disponível em: <<http://www.cin.ufpe.br/~tg/2012-2/jgff.pdf>>. Acesso em: 15 julho de 2013.

GOMES, R. C.; FERNANDES, J. A. R.; FERREIRA, V. C. **Sistema Operacional *Android***. 2012. Disponível em: <<http://www.midiacom.uff.br/~natalia/2012-1-sisop/tgrupol.pdf>>. Acesso em: 20 dez. de 2013.

GONÇALVES, J. C. **Uso da Plataforma *Android* em um Protótipo de Aplicativo Coletor de Consumo de Gás Natural**. 2011. Disponível em: <http://www2.dainf.ct.utfpr.edu.br/esp/monografias-de-especializacao-da-turma-vi-2010-2011/CT_JAVA_VI_2010_10.PDF/at_download/file>. Acesso em: 11 out. de 2013.

GOOGLE. ***quadrant standard***. GOOGLE PLAY, 2012. Disponível em: <https://play.google.com/store/apps/details?id=com.aurorasoftworks.quadrant.ui.standard&hl=pt_BR>. Acesso em: 15 nov. de 2013.

HAMANN, R. 2011. Disponível em: <<http://www.tecmundo.com.br/touchscreen/8091-as-telas-touchscreen-podem-estar-chegando-ao-fim-saiba-por-que.htm>>. Acesso em: 20 jan. de 2014.

HUBSCH, E. **Uma Abordagem Comparativa do desenvolvimento de aplicações para dispositivos móveis (Gerenciamento de memória)**. 2012. Disponível em: <<http://www.fatecsp.br/dti/tcc/tcc00065.pdf>>. Acesso em: 07 julho de 2013.

LACERDA, W. S. **Uma CPU simples para fins didáticos**. 2000. Disponível em: <http://www.dcc.ufla.br/infocomp/index.php?option=com_content&view=article&id=198&Itemid=73>. Acesso em: 02 dez. de 2013.

MARTINS, R. J. W. A. **Desenvolvimento de Aplicativo para *Smartphone* com a Plataforma *Android***. 2009. Disponível em: <<http://www.icad.puc-rio.br/~projetos/android/files/monografia.pdf>>. Acesso em: 20 nov. 2013.

MOBILEGAMER. **Site faz seleção dos apps e jogos mais populares do *Android***. 2013. Disponível em: <www.mobilegamer.com.br/2013/09/site-faz-selecao-dos-apps-e-jogos-mais-populares-do-android.html>. Acesso em: 15 jan. de 2014.

MORIBE, V. A. **Jogo para *Android* com *Unity3D***. 2012. Disponível em: <<http://fatecindaiatuba.edu.br/reverte/index.php/revista/article/view/73>>. Acesso em: 04 fev. 2014.

MOZARIK. **Qtadb**. 2013. Disponível em: <<http://qtadb.wordpress.com/>>. Acesso em: 15 agos. de 2013.

MUNIZ, L. M. **Gerenciamento de Processos e Memória do Sistema Operacional Android (Gerenciamento de memória)**. 2012. Disponível em: <<http://s3.amazonaws.com/ppt-download/temp-130527200307phpapp01.pdf?responsecontentdisposition=attachment&Signature=%2B6SVPu%2FV4r52JIquelMabI46d%2BI%3D&Expires=1373745857&AWSAccessKeyId=AKIAIW74DRRRQSO4NIKA>>. Acesso em: 07 julho de 2013.

PITOMBEIRA, D. K. D. **Uma Arquitetura Eficiente para Armazenamento, Gerenciamento e Acesso a Dados em Dispositivos Móveis com Recursos Computacionais Limitados**. 2006. Disponível em: <www.icad.puc-rio.br/~projetos/android/files/monografia.pdf>. Acesso em: 06 out. de 2013.

QTADB. **QtADB**. 2013. Disponível em: <<http://qtadb.wordpress.com/about/>>. Acesso em: 15 dez. de 2013.

RESEARCH, Juniper. **O mercado de jogos móveis em smartphones**. 2013. Disponível em: <<http://www.juniperresearch.com/reports.php?id=534>>. Acesso em: 20 agos. 2013.

ROVIO. **angry birds rio**. 2013. Disponível em: <<http://www.rovio.com/en/our-work/games/view/6/angry-birds-rio>>. Acesso em: 15 jan. de 2014.

SILVA, L. E. P. **Utilização da Plataforma Android no desenvolvimento de um aplicativo para cálculo de Balanço Híbrido**. 2013. Disponível em: <<http://pt.scribd.com/doc/109108976/TCC-Final>>. Acesso em: 20 nov. de 2013.

STALLINGS, William. **Arquitetura e organização de computadores: projeto para o desempenho**. In: _____. [S.l.: s.n.], 2010.

UFAM. **Alocador Slab**. 2013. Disponível em: <<http://sommil.wikispaces.com/Alocador+Slab#c>>. Acesso em: 15 jan. de 2014.