Universidade Federal de Itajubá
Campus Itabira
Engenharia de Computação

Usando Processing para desenvolvimento de aplicativos Android

Walter Aoiama Nagai

walternagai@unifei.edu.br

ERI 2016 – 18 de Novembro de 2016

Quem sou eu?

- **Walter Aoiama Nagai**
- **Formação**
  - Mestre em Ciências de Computação e Matemática Computacional – ICMC/USP-São Carlos
  - Bacharel em Ciência da Computação – UFMS
- **Experiências profissionais**
  - Universidade Federal de Itajubá – 2010-atual
  - Centro Universitário Barão de Mauá – 2000-2009
  - Universidade de Ribeirão Preto (UNAERP) – 2007-2009
- **Experiências em desenvolvimento**
  - C/C++, Java, PHP, HTML5, CSS, Javascript
  - Diversos bancos de dados

Sumário

We Are Social

Apresentação

- Agência global para coleta de dados, estatísticas, tendências para entendimento o estado atual das mídias digitais, sociais e móveis ao redor do mundo – `http://wearesocial.com/`.
- Agência brasileira: `http://wearesocial.com/br/`
- Relatório de Tendências Digitais em 2016 para o Brasil `http://wearesocial.com/uk/special-reports/digital-in-2016`, Brasil, página 95

Vendas Globais por Sistemas Operacionais

Apresentação

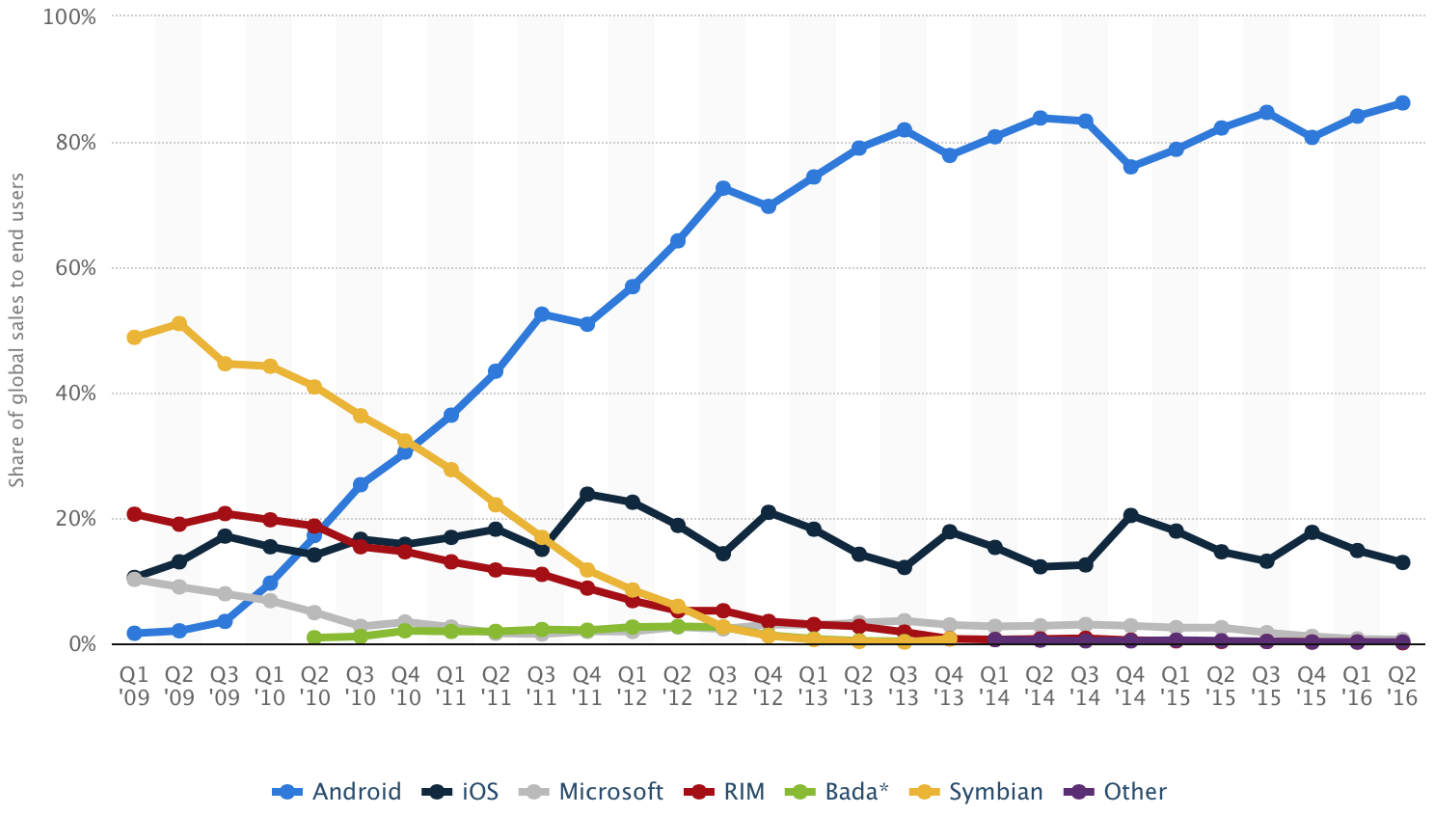


Figura: Vendas globais de sistemas operacionais móveis de 2009 à 2016.
FONTE: <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>

Dificuldades

Apresentação

- Mercado dominado por duas empresas: Apple & Alphabet (Google)
- Linguagens e *frameworks* diferentes
- *Integrated Development Environment* (IDE) diferentes
- Interações e usabilidades diferentes
  - Integridade Aestética – `https://developer.apple.com/ios/human-interface-guidelines/`
  - *Material Design* – `https://material.google.com`
- Computação Criativa [Reas and Fry 2015], independentemente do ambiente que são executados, os aplicativos devem ter uma interface interativa e suscetível a rápidas interpretações por parte dos usuários que os utilizam.

Arquitetura do *Android*

Linux Kernel

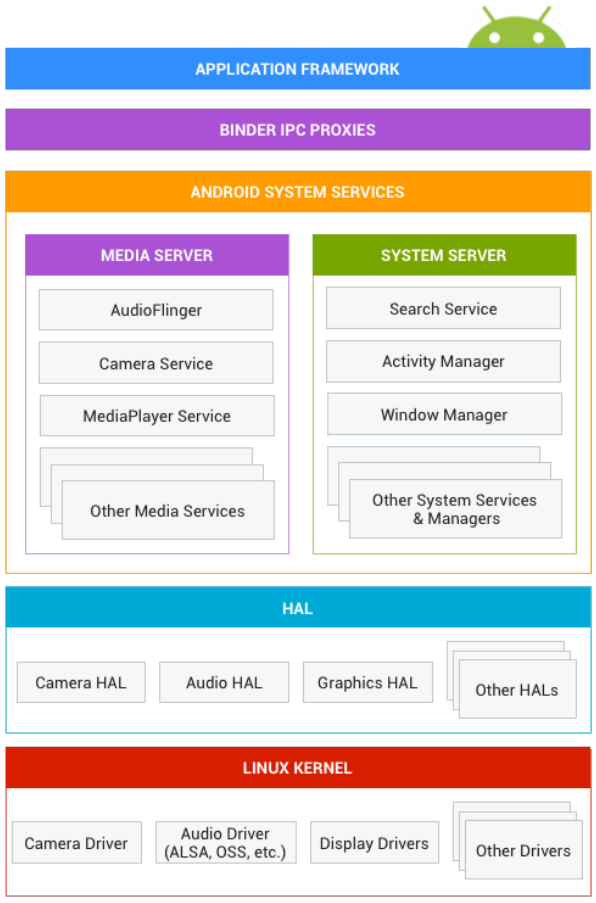


Figura: Arquitetura do sistema operacional móvel *Android*.
FONTE: <http://source.android.com/devices/index.html>

Na *Linux Kernel*, encontra-se o *kernel* ou núcleo do um sistema operacional Linux, versão 4.4, e as implementações de drivers para dispositivos essenciais do sistema.

Arquitetura do *Android*

HAL

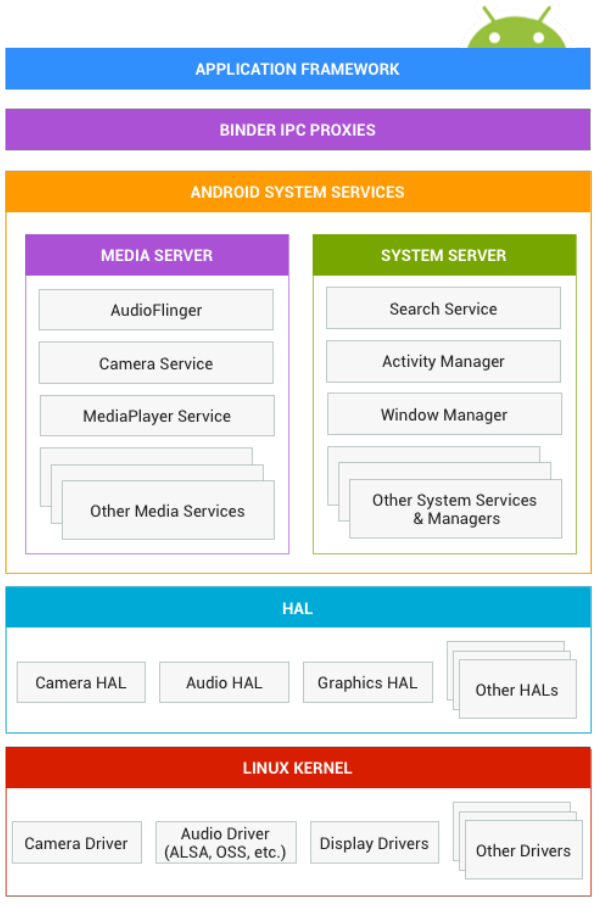


Figura: Arquitetura do sistema operacional móvel *Android*.
FONTE: <http://source.android.com/devices/index.html>

O *Hardware Abstraction Layer* (HAL) é a parte do sistema operacional que estabelece uma interface de comunicação para drivers específicos com o kernel do sistema operacional.

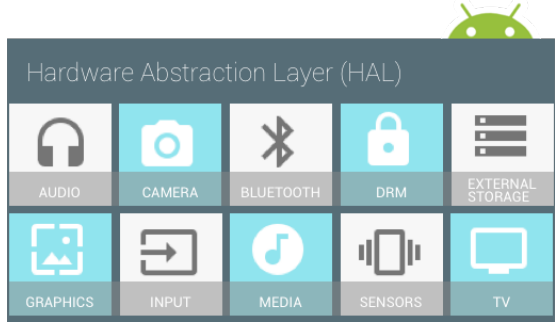


Figura: HAL do sistema operacional *Android*.
FONTE: <http://source.android.com/devices/index.html>

Arquitetura do *Android*

Android System Services

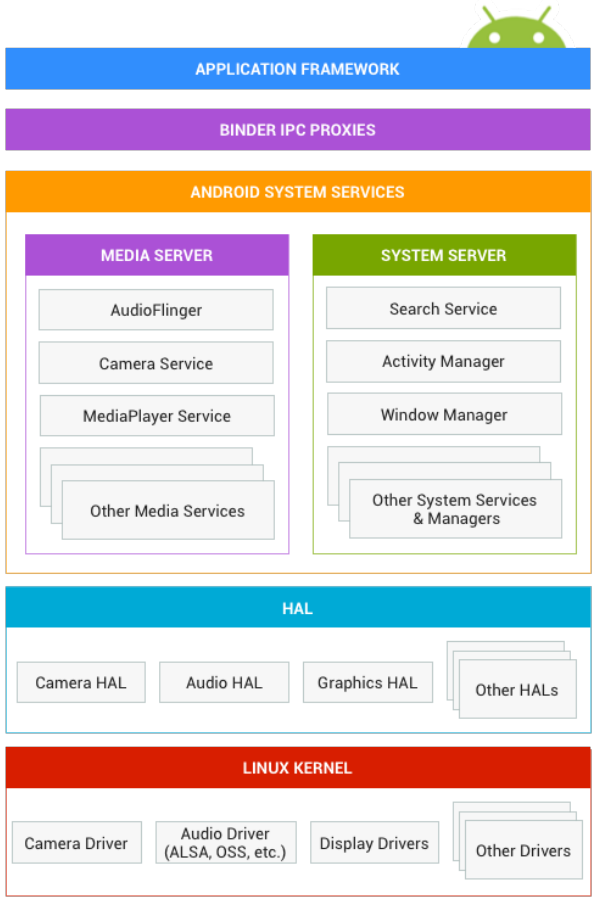
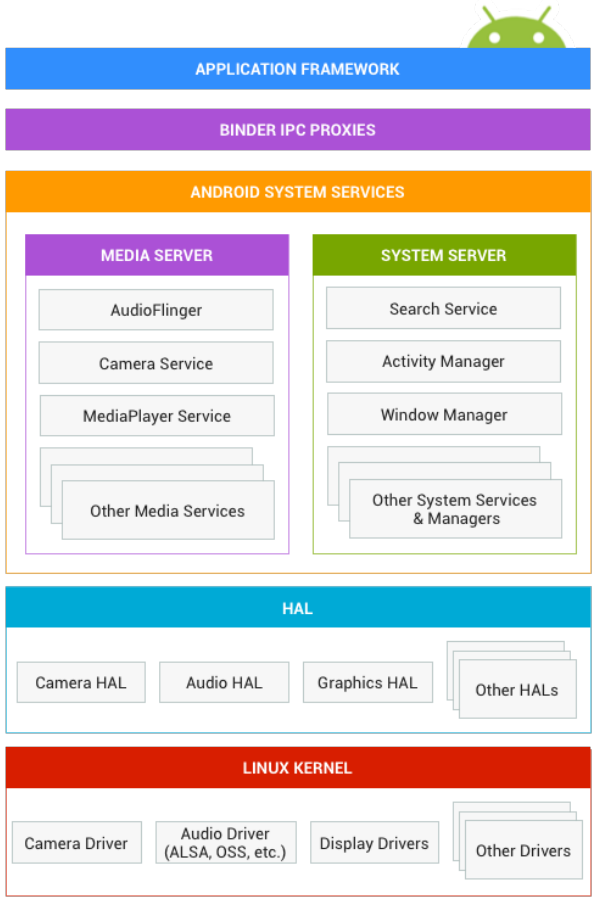


Figura: Arquitetura do sistema operacional móvel *Android*.
FONTE: <http://source.android.com/devices/index.html>

Nesta parte por meio de *Application Programming Interface* (API) os serviços acessam as partes mais baixas do sistema e são divididos em dois grupos: (i) *system* como *Window e Notification Managers* e (ii) *media* para reproduzir e gravar mídias digitais.

Arquitetura do *Android*

Binder IPC Proxies

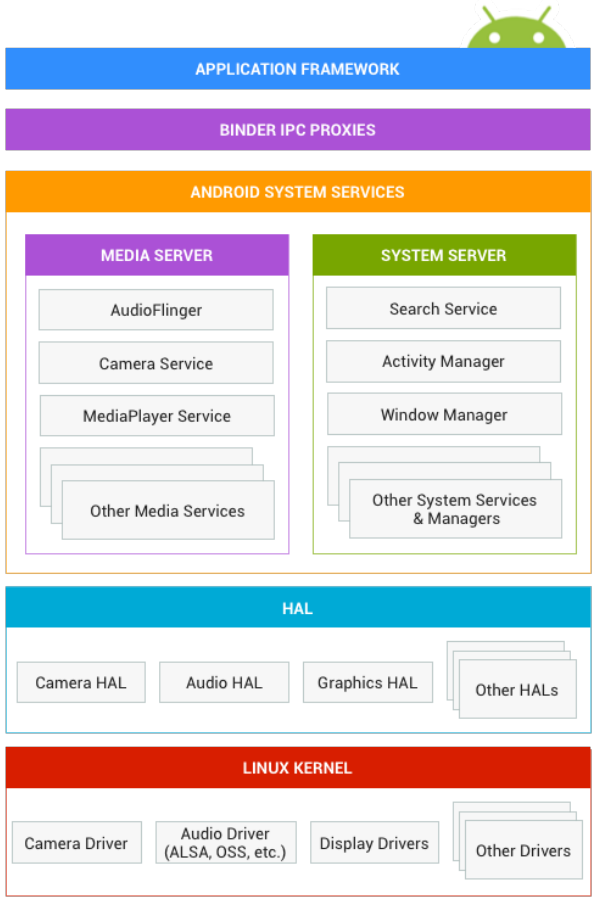


A parte permite que os aplicativos desenvolvidos em determinado API possam interagir com códigos dos serviços disponíveis do sistema *Android*.

Figura: Arquitetura do sistema operacional móvel *Android*.
FONTE: <http://source.android.com/devices/index.html>

Arquitetura do *Android*

Application Framework



A camada do sistema operacional que estão disponíveis o conjunto de *frameworks*, bibliotecas e códigos para permitir uma correta execução de um aplicativo *Android*.

Figura: Arquitetura do sistema operacional móvel *Android*.
FONTE: <http://source.android.com/devices/index.html>

IDE *Android Studio*

- Para o desenvolvimento de aplicativos no *Android* é utilizado o IDE *Android Studio*
  - O `Instant Run` analisa as alterações no código do aplicativo e, se necessário, as alterações já são inseridas no aplicativo em tempo de execução
  - O `Layout Editor` melhora a visualização das telas e as interações entre elas;
  - O `Emulator` executa as máquinas virtuais criadas para cada versão do *Android*.

Instruções DEX

- Um aplicativo é convertido em uma sequência de códigos para uma máquina virtual baseada em registradores com instruções *Dalvik Executable Format* (DEX). O *Dalvik* foi, por muitos anos, o software responsável em executar as instruções DEX de forma segura e completa de um aplicativo.
- A partir do Android 4.4 Kit Kat, API 19, surgiu o *Android Run-Time* (ART) que trouxe várias inovações e melhorias para as instruções DEX.

Componentes *Android*

1. **Atividade** ou *Activity* representa uma tela única com uma interface do usuário;
2. **Serviços** ou *Services* são componentes executados em segundo plano (*background*) para realizar operações longas de execução ou para realizar trabalhos de processos de modo remoto;
3. **Provedores de conteúdo** ou *Content Providers* gerenciam um conjunto compartilhado de dados de um aplicativo;
4. **Receptores de transmissão** ou *Broadcast Receivers* são componentes que respondem a anúncios de transmissão por todo o sistema.

Activity

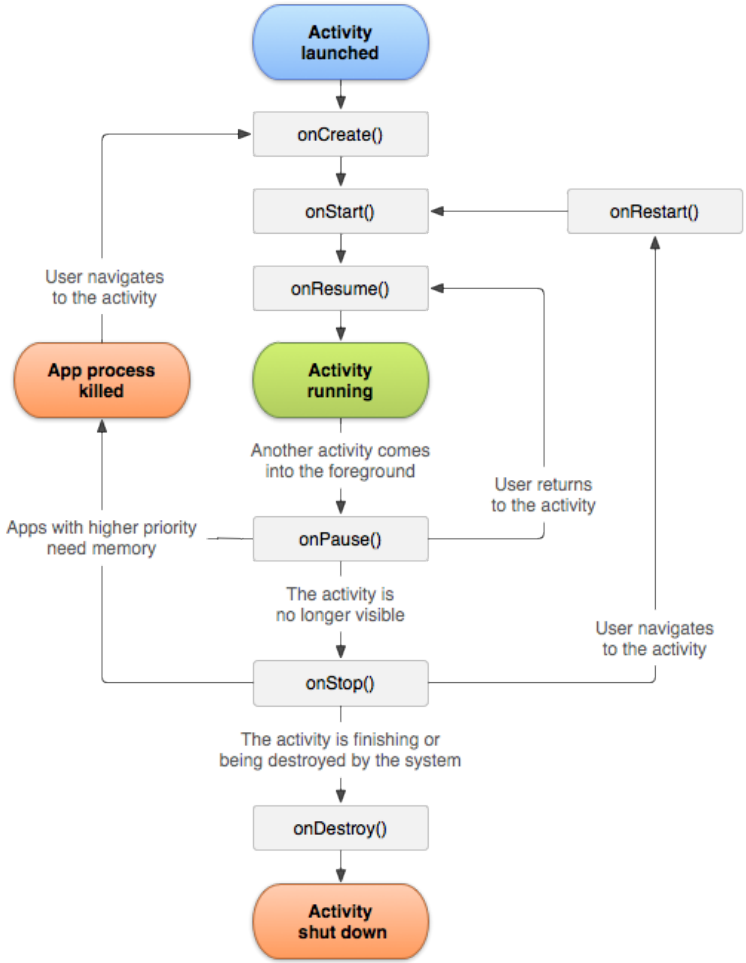


Figura: Ciclo de vida de uma Atividade no sistema *Android*.
FONTE:
<https://developer.android.com/reference/android/app/Activity.html>

IDE *Processing*

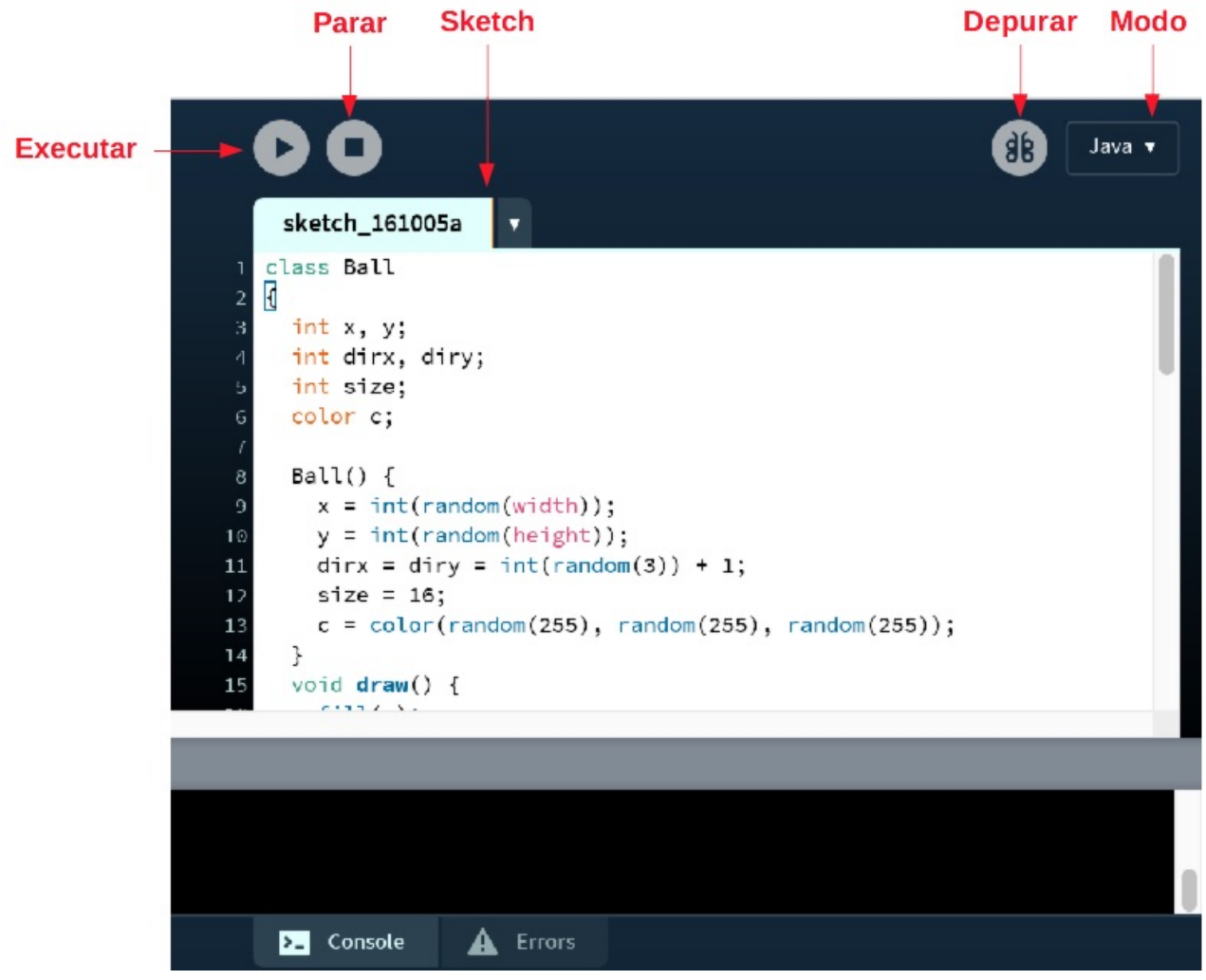


Figura: Janela principal do IDE *Processing*. FONTE: <http://www.processing.org>

sketch myBall

```
Ball myBall[];
int amount;

void setup()
{
  fullScreen();
  amount = int(random(100)) + 1;
  myBall = new Ball[amount];
  for (int i = 0;
       i < amount; i++)
    myBall[i] = new Ball();
}
```

```
void draw()
{
  background(0);
  fill(255);
  for (int i = 0; i < amount; i++)
  {
    myBall[i].update();
    myBall[i].checkCollision();
    myBall[i].draw();
  }
}
```

sketch Ball

```
class Ball {
  int x, y, dirx, diry, size;
  color c;
  Ball() {
    if(width > height) this.size = height/16;
    else this.size = width/16;
    this.x = int(random(width/2)) + this.size;
    this.y = int(random(height/2)) + this.size;
    this.dirx = this.diry = int(random(10)) + 1;
    this.c = color(random(255), random(255), random(255));
  }
  void draw() {
    fill(this.c);
    ellipse(this.x, this.y, this.size, this.size);
  }
  void update() {
    this.x += this.dirx;
    this.y += this.diry;
  }
  void checkCollision() {
    if (this.x < size/2 || this.x > width-size/2) this.dirx *= -1;
    if (this.y < size/2 || this.y > height-size/2) this.diry *= -1;
  }
}
```

PARTE II

Usos do *Processing*

- Utilizar o próprio IDE *Processing* para desenvolver aplicativos *Android*;
- Utilizar o IDE *Android Studio* para desenvolver aplicativos e "embutir" códigos descritos em *Processing*.

Desenvolvendo aplicativos *Android* usando o IDE *Processing*

- conecte um *smartphone Android* 6.0 *Marshmallow*, API 23 com o próprio cabo USB da fabricante;
- habilite o modo de depuração de aplicativos *Android*;
- insira os *sketch* `myBall` e `Ball` no IDE *Processing*;
- altere o modo `Java` para o modo `Android`;
- clique no botão `Run` ou `Executar`;
- por último, o aplicativo *Android* produzido será copiado para o *smartphone* e executado.

Classe PApplet

- A classe `PApplet` é instanciada em um `Fragment` ou fragmento.
- A partir do *Android Honeycomb*, API 11, o Google introduziu o conceito de Fragmento ou *Fragment*, que segundo [Android 2016a]: "*Um fragmento é como uma seção modular de uma atividade, que tem o próprio ciclo de vida, recebe os próprios eventos de entrada e que pode ser adicionado ou removido com a atividade em execução*".

Fragment

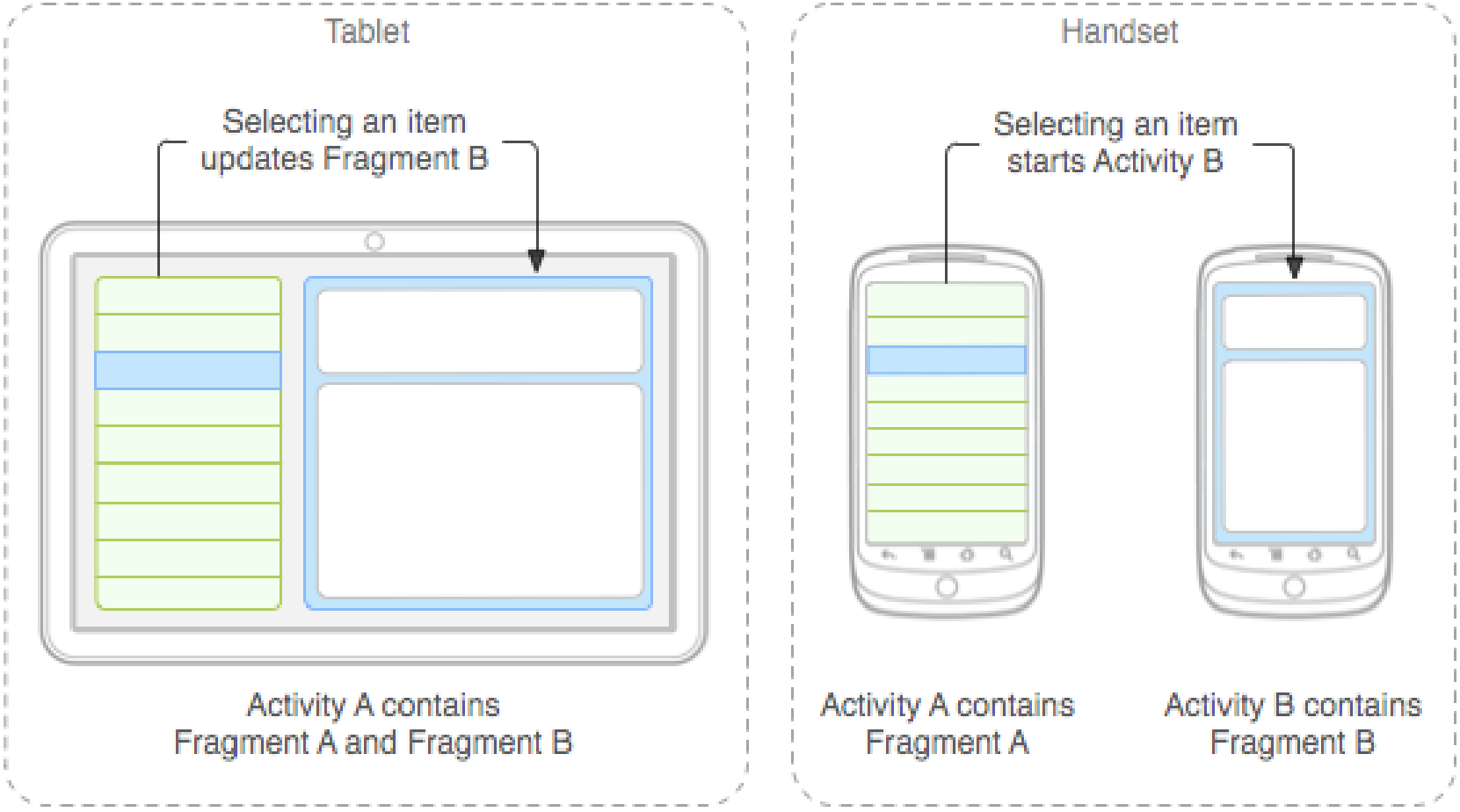


Figura: Uso de dois `Fragment` em uma única Atividade e um `Fragment` em cada Atividade.
FONTE: <https://developer.android.com/guide/components/fragments.html>

Desenvolvendo aplicativos usando o *Android Studio* com códigos em *Processing*

- Crie um novo projeto no *Android Studio*;
- no campo "`Application name`", digite `FragExample`;
- no campo "`Company domain`", digite `br.edu.unifei.labsmart` e depois clique em `Next`;
- escolha a plataforma e selecione em "`Minimum SDK`", `API 23: Android 6.0 (Marshmallow)` e em seguida, clique em `Next`;
- escolha a Atividade, "`Empty Activity`" e clique em `Next`;
- na tela "`Customize the Activity`" não é necessário alterar nenhum campo. Clique em `Next` e espere pelo *Android Studio* criar todos os arquivos necessários, configurar todos os *builds*, etc.

Desenvolvendo aplicativos usando o *Android Studio* com códigos em *Processing*

- Procure na pasta do *Processing*, a pasta `modes` e depois a pasta `AndroidMode`;
- copie o arquivo `android-core.zip` na pasta do projeto do *Android Studio*, `AndroidStudioProjects/FragExample/app/libs`, renomeando-o para `android-core.jar`;
- no menu "`File`", escolha a opção `Project Structure...`;
- clique em "`app`" e depois em "`Dependencies`";
- clique no ícone + e escolha a opção "`2 File Dependency`". Isso irá abrir todas as pastas do projeto `FragExample`;
- abra a pasta "`libs`" e clique em `android-core.jar`;
- crie uma nova classe `MyBall` e adicione o código do `MyBall.java`;

Desenvolvendo aplicativos usando o *Android Studio* com códigos em *Processing*

Altere o código da classe `MainActivity`, adicionando no método `onCreate`, o seguinte trecho de código:

```
FragmentManager fragmentManager = getFragmentManager();
        Fragment fragment = new MyBall();
        fragmentManager.beginTransaction()
                .replace(R.id.container, fragment)
                .commit();
```

Desenvolvendo aplicativos usando o *Android Studio* com códigos em *Processing*

Altere o arquivo *layout* `activity_main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=
      "labsmart.unifei.edu.br.fragexample.MainActivity">

    <FrameLayout android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

Class MyBall

```
package labsmart.unifei.edu.br.fragexample;
import processing.core.PApplet;

public class MyBall extends PApplet {
    Ball myBall[];
    int amount;
    public void settings() {
        size(800, 600);
    }
    public void setup() {
        amount = PApplet.parseInt(random(100)) + 1;
        myBall = new Ball[amount];
        for (int i = 0; i < amount; i++)
            myBall[i] = new Ball();
    }
    public void draw() {
        background(0); fill(255);
        for (int i = 0; i < amount; i++) {
            myBall[i].update();
            myBall[i].checkCollision();
            myBall[i].draw();
        }
    }
```

Class MyBall

```
    class Ball {
        int x, y, dirx, diry, size, c;
        Ball() {
            if(width > height) this.size = height/16;
            else this.size = width/16;
            this.x = PApplet.parseInt(random(width/2)) + this.size;
            this.y = PApplet.parseInt(random(height/2)) + this.size;
            this.dirx = this.diry = PApplet.parseInt(random(10)) + 1;
            this.c = color(random(255), random(255), random(255));
        }
        public void draw() {
            fill(this.c);
            ellipse(this.x, this.y, this.size, this.size); }
        public void update() {
            this.x += this.dirx; this.y += this.diry; }
        public void checkCollision() {
            if (this.x < size/2 || this.x > width-size/2)
              this.dirx *= -1;
            if (this.y < size/2 || this.y > height-size/2)
              this.diry *= -1;       }
    }
}
```

Considerações finais

- Em [Processing 2016b] podem ser encontrados alguns projetos adicionais de aplicativos *Android*;
- Uso do *Processing* é fácil para programadores novatos;
- Biblioteca `Ketai` disponível em `<http://ketai.org>` permite acesso a sensores, câmeras, componentes de rede *wireless*, etc., disponíveis em *smartphones*.
- No link `<https://github.com/wnagai/ERI-MT-2016>` estão disponíveis os códigos de alguns aplicativos desenvolvidos para o IDE *Processing* e para o IDE *Android Studio*.

Contato

Walter Aoiama Nagai

- **Endereço**
  Rua Irmã Ivone Drummond, 200
  Distrito Industrial II
  Itabira - MG, 35903-087
- **E-mail**
  `walternagai@unifei.edu.br`
- **Grupos de Pesquisa**
  Computação Aplicada à Engenharia -
  `https://gpcae.unifei.edu.br`
  Metodologias Ativas para o Ensino Superior -
  `https://maes.unifei.edu.br`
- **Agradecimentos**
  UNIFEI - Campus Itabira.

Referências I


Android (2016a).
Fragmentos.
Disponível em:
<https://developer.android.com/guide/components/fragments.html>.


Android (2016b).
Fundamentos.
Disponível em:
<https://developer.android.com/guide/components/fundamentals.html>.


DEVMedia (2015).
Ciclo de vida do fragments no android.
Disponível em: <http:
//www.devmedia.com.br/ciclo-de-vida-do-fragments-no-android/33099>.


Google (2016).
Developer android.
Disponível em: <http://developer.android.com>.


Greenberg, I., Xu, D., and Kumar, D. (2012).
Processing: creative coding and generative art in Processing 2.
Apress Inc.


Processing (2016a).
Processing.
Disponível em: <http://www.processing.org>.

Referências II


Processing (2016b).
Processing for android.
Disponível em: <http://android.processing.org>.


Reas, C. and Fry, B. (2015).
Make: Getting started with Processing.
Maker Media Inc., 1160 Battery Street East, Suite 125, San Francisco, CA 94111.


Runberg, D. (2015).
The Sparkun Guide to Processing: create interactive art with code.
William Pollock, 245 8th Street, San Francisco, CA 94103.


Shiffman, D. (2008).
Learning Processing: a beginners guide to programming images, animation and interaction.
Morgan Kaufmann Publishers, 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA.


Shiffman, D. (2012).
Nature of Code.
On my own, Disponível em: <http://natureofcode.com/book/>.