Droid Fit Android における UI 設計支援ライブラリ

Droid Fit: A library to support user interface design for Android

永橋洋明 †, 福田浩章 ‡
Hiroaki Nagahashi†, and Hiroaki Fukuda‡
† 芝浦工業大学　工学部　情報工学科
‡ 芝浦工業大学　分散ソフトウェアシステム研究室
†Faculty of Information, Shibaura Institute of Technology Univ.
‡Distributed Software System, Shibaura Institute of Technology Univ.

要旨

メーカーによって端末の仕様が異なる Android スマートフォンが急増する中, 端末依存の問題は避けられない. 問題はアプリケーションのレイアウト設計にも及んでおり, 対応するためにはレイアウトファイルを複数用意する必要がある. テスト端末を 1 台しか所持していない場合, 充分な確認ができず問題に対応できない Android アプリケーション開発者も少なくない.

本研究では, 開発時の端末の情報を基に, アプリケーション実行時に拡縮比率を計算し, アプリケーションのレイアウトサイズをスケールするライブラリを提案する.

1. 研究の背景と目的

世界中のスマートフォンの割合は, Android と iOS が大多数を占めている [1]. iOS は初代 iPhone をはじめ, 12 種類のスマートフォンが存在する. 一方 Android は 2015 年時点で 24093 種類にも及ぶスマートフォンが存在する [2]. 解像度についても, iOS は数種類であるのに対し, Android は非常に多くの種類が存在する. Android アプリケーションを開発する際, この多くの種類の解像度に対応しなければならない. そのため対応には多くのテスト端末が必要であるが, 個人の開発者では充分な端末数を用意するのは非常に困難であり, その結果レイアウト関連の対応が不完全のまま公開されることも少なくない.

さらに近年では Android5.0 の普及と同時にマテリアルデザインが Google によって定められた [3]. アプリケーションを統一されたデザインにすることはユーザにとっては操作性の向上に繋がるが, 開発者にとってはデザインの変更, 場合によってはデザインに適した機能の修正が必要になる.

そこで本研究では, 開発端末の情報を基に実行端末でレイアウトをスケール表示する機能, 色, フォントサイズ, 画面を構成する部品である View のサイズをマテリアルデザインで指定されているデザインへ変更する機能を備えた, UI 設計支援ライブラリ DroidFit を提案する.

2.Android における現状

2.1. レイアウト問題

レイアウト配置の不具合は, レイアウトを作成する際に静的な値を指定しまう場合に起こりやすい.

Android では一般的に用いられる pixel(px) のほか, Density-independent pixels（dp）が使用されている. dp は, Android の特徴である多彩なピクセル密度 (DPI) に対応するための単位である. DPI を 160 で除算した Density という値によって, px と dp の関係を保持している. Android ではレイアウトファイルに dp の固定値を記述した場合, Density や解像度の違いによって開発者が想定したレイアウト通りに表示されない場合がある.

2.2. マテリアルデザイン問題

マテリアルデザインとは Google が 2014 年 6 月に発表したデザインであり, Android5.0 以降はこのデザインを適用することが望ましいとされている. ガイドラインではアニメーション, 色, アイコンのデザインの仕方, フォントサイズ, 画面のレイアウト配置などが示されており, これらを以前のアプリケーションに適用するには多くの工数が発生してしまう.

3.Droid Fit による問題解決

3.1. スケール機能の実現

Density の違いはレイアウトファイルを DPI の値ごとに用意することで既に解決されている. Density が同じ場合は解像度の差によってレイアウト配置の不具合が発生する. 例えば Density が 2, 解像度が

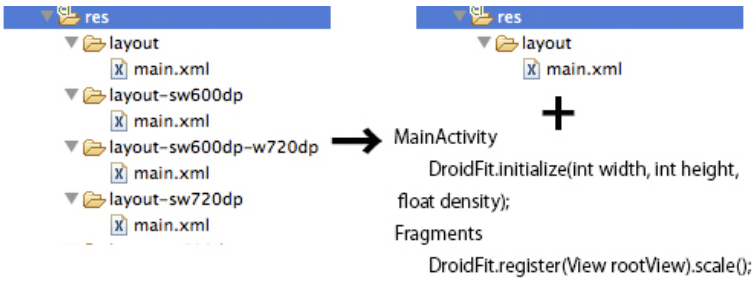

図 1: ライブラリ導入前後

480x800px の端末に対し, Density は同値で解像度が 960x1600px の端末で比較した場合, 縦横それぞれ 2 倍に拡大されるべきである. しかしレイアウトファイルで dp を使用した場合, 後者は前者と同様のピクセル数で表示される. これは開発者が想定した端末の Density 及び解像度を元に, アプリケーション実行端末で表示されるべき値を計算することで補正することができる. Density 及び解像度の比率を計算し, 求められた実際のサイズを各 View に適用することで, 問題を解決することができる. Droid Fit では, プログラムで 3 つのメソッドを実行することにより計算及び View への適用を行う. その結果, 導入以前は各 DPI ごとにレイアウトファイルを作成する必要があったが, 導入後はレイアウトファイルを削減し, メソッド呼び出しを数行追加するのみである (図 1). 実際の計算式は以下である. (各数字は 1 が開発端末, 2 が実行端末, w は width, h は height)

$$RealBeforePixel : rbpx = dp * density2 \tag{1}$$

$$DensityScale : ds = density1/density2 \tag{2}$$

$$ResolutionScale : rs = min(w2/w1, h2/h1) \tag{3}$$

$$RealAfterPixel : rapx = rbpx * ds * rs \tag{4}$$

3.2. マテリアルデザイン適用機能の実現

マテリアルデザインガイドラインにおいて, 色, フォントサイズ, 各 View への対応には一定のパターンを持つ部分が存在する. 色はガイドライン上で 256 種類が指定されており, 2 点間距離の計算によって近似した色を発見, 適用する. フォントサイズはタイトルやボタンなど 11 種類に分類されており, クラス比較やリフレクションを用いて対象を発見する. 各 View への対応は, ボタンや最外部のマージンの数値が指定されており, DroidFit に内包したプリセットを参照して適用する. そのパターンの適用をメソッド化し自動化することで, 開発者の負担を減らすことができる. Droid Fit ではガイドラインに準拠したプリセットをライブラリに内包させ, 対応メソッドを呼び出すことで, 対応した View を継承したクラスに適用する.

4. 設計と実装

4.1. スケール機能

スケール機能では, 次のメソッドを提供する.

- initialize(Context context, DeviceInfo testDevice) Android アプリケーション実行クラスである Activity の情報, テスト端末の情報を登録
- register(View rootView) スケールする最上位 View の登録
- scale() スケールの実行
- setScaleFor(Verifier verifier,View... views) スケール対象とする View とその条件の登録
- setUnScaleFor(Verifier verifier,View... views) 非スケール対象とする View とその条件の登録

initialize, register 及び scale メソッドを実行することで, テスト端末の情報を基に register で登録した最上位 View から再帰的にスケールされる．register メソッドを実行後, setScaleFor, setUnScaleFor メソッドを使用することで特定の View をスケール対象外にすることが可能である．

4.2. マテリアルデザイン適用機能

マテリアルデザイン適用機能では, 次のメソッドを提供する．

- register(View rootView) マテリアルデザイン化する最上位 View の登録
- material() マテリアルデザインの適用
- setMaterialFor(Verifier verifier,View... views) マテリアルデザイン化対象とする View とその条件の登録
- setUnMaterialFor(Verifier verifier,View... views) 非マテリアルデザイン化対象とする View とその条件の登録

register 及び material メソッドを実行することで, register で登録した最上位 View から再帰的にマテリアルデザイン化される．register メソッドを実行後, setMaterialFor, setUnMaterialFor メソッドを使用することで特定の View をマテリアルデザイン化対象外にすることが可能である．

5. まとめと今後の予定

現在スケール機能の実装が終了している．今後は, ライブラリ本体の作成, テストに使用するアプリケーションの作成を最初に行い, 実際に AWS Device Farm を使用してテストを行う [4].

参考文献

[1] ”Android and iOS Squeeze the Competition, Swelling to 96.3% of the Smartphone Operating System Market for Both 4Q14 and CY14, According to IDC,” IDC, http://www.idc.com/getdoc.jsp?containerId=prUS25450615, 09/25/2015.

[2] ”Android Fragmentation Visualized,” OpenSignal, http://opensignal.com/reports/2015/08/android-fragmentation/, 09/25/2015.

[3] ”Material design Guideline,” Google, http://www.google.com/design/spec/material-design/introduction.html, 09/25/2015.

[4] ”AWS Device Farm,” Amazon Web Services, http://aws.amazon.com/jp/device-farm/, 09/25/2015.