Wstęp do Material Design (IV)

Marek Wojtuszkiewicz

Modyfikacja ImageView

```java
public class OneOneImageView extends ImageView {

    public OneOneImageView(Context context) { super(context); }

    public OneOneImageView(Context context, AttributeSet attrs) { super(context, attrs); }

    public OneOneImageView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int height = MeasureSpec.getSize(widthMeasureSpec);
        int heightSpec = MeasureSpec.makeMeasureSpec(height, MeasureSpec.EXACTLY);
        super.onMeasure(widthMeasureSpec, heightSpec);
    }
}
```

RoundedBitmapDrawable

Zmiana „kształtu” bitmapy na okrągły

```
RoundedBitmapDrawable drawable = RoundedBitmapDrawableFactory
        .create(getResources(), bitmap);
drawable.setCircular(true);
```

RippleDrawable

Dodaje efekt „fali” w odpowiedzi na zdarzenie dotyku

Klasa dziedziczy po LayerDrawable

```
<ripple xmlns:android="http://schemas.android.com/apk/res/andro
    android:color="?android:colorControlHighlight">
    <item>
        <shape android:shape="rectangle">
            <solid android:color="@color/colorAccent" />
            <corners android:radius="2dp" />
        </shape>
    </item>
</ripple>
```

StateListAnimator

Animuje zmiany stanu wskazanego widoku

```xml
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_enabled="true"
        android:state_pressed="true">
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="8dp"
            android:valueType="floatType" />
    </item>
    <item>
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="0dp"
            android:valueType="floatType" />
    </item>
</selector>
```

Efekt „circular reveal”

```
Animator anim = ViewAnimationUtils.createCircularReveal
        (view, view.getWidth() / 2, view.getHeight() / 2, 0, revealRadius);
view.setBackgroundColor(Color.GRAY);
anim.start();
```

Ekstrakcja kolorów

- Vibrant
- Vibrant Dark
- Vibrant Light
- Muted
- Muted Dark
- Muted Light

```
Palette palette = Palette.from(bitmap).generate();

// lub

Palette.from(bitmap).generate(new Palette.PaletteAsyncListener() {
    @Override
    public void onGenerated(Palette palette) {
        // kod wykonany asynchronicznie
    }
});
```

Wstęp do Material Design (IV) - zadanie

Rozszerz aplikacją utworzoną w poprzednim zadaniu:

1. Zmodyfikuj widok ImageView na potrzeby nagłówka ekranu i użyj go na ekranie „O aplikacji” (https://material.io/guidelines/layout/metrics-keylines.html)
2. W nagłówku widoku NavigationView wyświelt avatar użytkownika zgodny z Material Design pod względem kształtu i rozmiaru ze źródłowej bitmapy będącej prostokątem (https://material.io/guidelines/layout/metrics-keylines.html)
3. W jednym z fragmentów utwórz widok typu CardView (domyślnie niewidoczny) oraz przycisk (domyślnie widoczny)
    1. Po wciśnięciu przycisku widok karty powinien się pojawić wraz z efektem „circular reveal”
    2. Po wciśnięciu (i trzymaniu) karty, jej wysokość nad tłem powinna zostać zwiększona oraz powinien pojawić się efekt fali
    3. Po wciśnięciu karty powinna ona się schować wraz z odwrotnym efektem „circular reveal”
    4. Karta powinna zawierać zdjęcie, tytuł, podtytuł oraz dłuższy tekst
    5. Kolorystyka karty powinna być dostosowana dynamicznie do wyświetlanego na niej obrazka (przykład: https://storage.googleapis.com/material-design/publish/material_v_10/assets/0Bzhp5Z4wHba3am9VWWFlbHVJNDg/components_cards3.png)
    6. Kształt, style tekstu i rozmieszczenie elementów na karcie powinno być w pełni zgodne z zasadami Material Design (https://material.io/guidelines/components/cards.html)