Android Ninja Series

Edward Pie

Foreword

Android Is Everywhere, And Android Has Come To Stay For Good

The Android mobile operating system has come to stay for our good. It present enormous opportunities for developers and businesses of all sizes. Already, Android powers millions of mobile devices (phones, tablets, wearable etc) and a million more of such devices are activated each day. This means developers can target more prospective audiencec by building apps that change how people live, communicate, transact business, purchase goods etc.
A larger prospective targets audience translates directly into a larger market with a potential of making millions of cash. The rate of growth of use of Android devices is exponential.

Android devices are fitted with useful sensors such as microphones, cameras, accelerometers, gyroscopes, magnetic field sensors and many more. With these sensors, developers can sense continuously varying quantities in nature. In effect, Android devices are environment-aware. They can sense their environment and take actions based on the data collected.

The Google play store and other Android app stores give every developer a ticket to the game, an equal opportunity to make money in bits. The Google play store is flooded with apps in the categories of games, entertainment, lifestyle, health, education, religion, utilities and others. With just about $25.00, everyone can build and publish apps onto the app store for people to download for free or purchase.

What's more fun? All the tools used for developing Android applications are completely free. I mean free of charge. Everything from the Android SDK, to IDEs such as Eclipse, Android Studio etc are all free. Additionally, there are other freely available online utilities for generating assets for use in Android apps.

The time to develop Android apps, is NOW!

About The Author

Edward Pie is a Computer Science major from the Kwame Nkrumah University Of Science And Technology in Kumasi, Ghana. He's a polyglot and a passionate programmer who does anything right, for as long as it takes to create a world where nothing is difficult and everything is simple. He loves to adhere strictly to high standards and abores mediocrity completely. He learns to teach and teaches to learn.
He's worked with Saya Mobile (which recently got acquired by Kirusa), IC Securities and Rancard Solutions Ltd. He has also worked on projects as a freelancer.

He loves mobile apps development, robotics and electronics.

Author's Contact Information

Edward Pie
hackstockpie@gmail.com
edwardpie.herokuapp.com
edwardpie.com
geekiepie@blogspot.com
+233200662782

About This Book

The purpose of this book is to present to it's readers, recipes on how-to get specific things done on the Android platform. It's not far from a cookbook except that each publication focuses on just one aspect of the Android platform. The aim it to be an on-point and sure way of getting things done professionally without bordering readers with unnecessary theory. Simply, this and many more of such publications from Edward Pie helps you get things done in the shortest possible time.

Using Code Examples

This book is here to help you get your job done. In general, if this book includes code examples, you may use the code in this book in your programs and documentation. You do not need to contact me for permission unless you've enough time to waste.
I appreciate, but do not require attribution.

All code samples and other publications can be downloaded from my public github repo. https://github.com/hackstock/ninja-series.git.

Building Splash Screens In Android

Splash screens are usually the first screens shown when an app is launched. Many applications perform resource initializations while the splash screen is shown.
A good splash screen is usually as good as your application's name. It can be a turning point for your users. A poorly designed splash screen can turn off your users. A good splash screen however, is aesthetically appealing to the eyes (ask your daddy), gives proper feedback to users and allows itself to be interrupted if necessary

The text in this publications is going to show you how to build splash screens in Android. The example used wouldn't be appealing to the eye but will teach you how.

Where To Find Inspirations For Your Designs

Not everyone is born with design skills. Usually, large development teams have members whose jobs is to design aesthetically beautiful UIs but as indie developers as we are, we are often required to do our own designs and trust me, it gets hairy at times.
I suck when it comes to graphics. I such big time but www.pttrns.com has been an invalualbe tool in my development arsenal since I discovered it. For this text, http://www.pttrns.com/categories/2-launch-screen will help you with some ideas on how to design a good splash screen for your app.

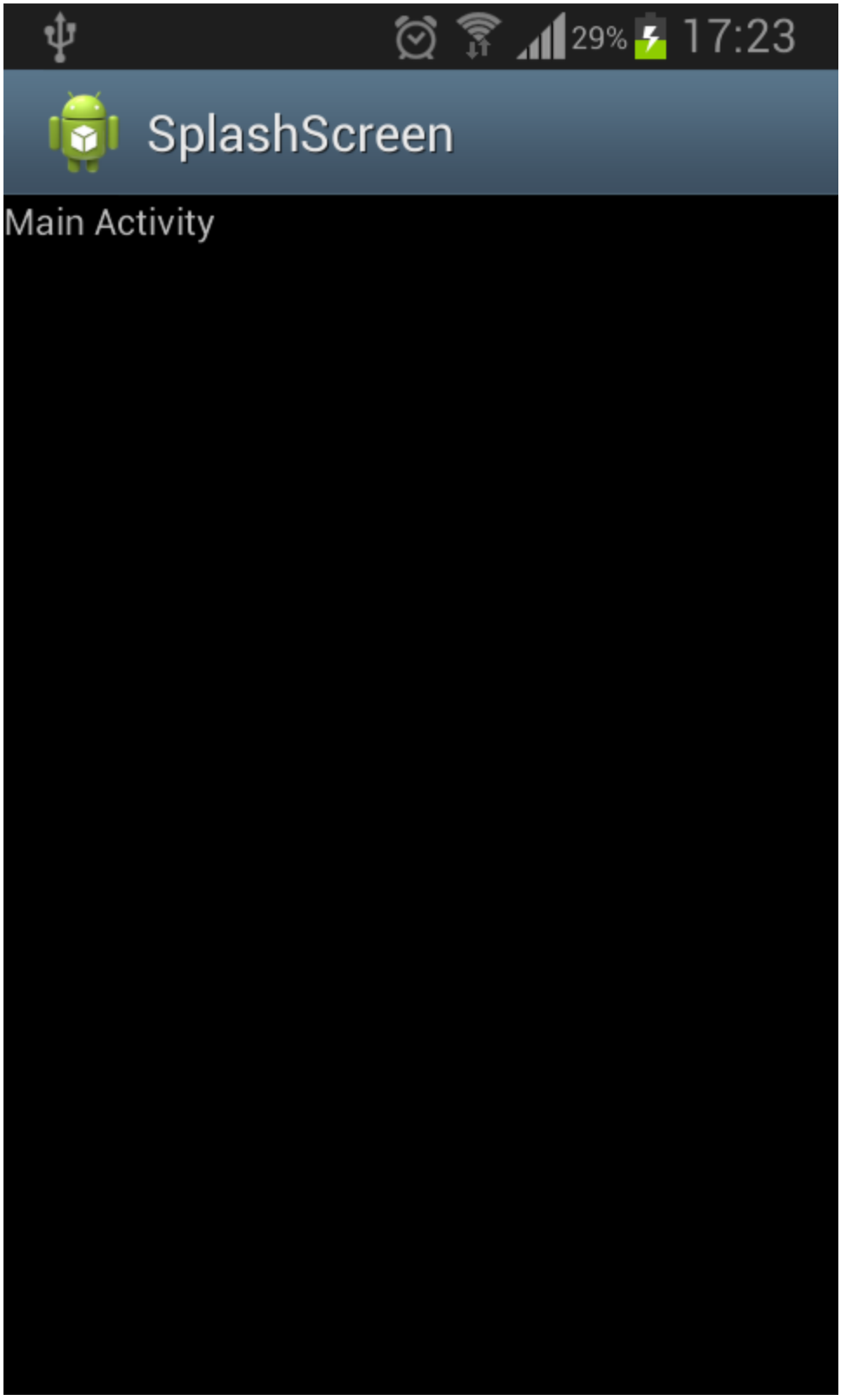
Our Awesome Splash Screen

The sample project used in this text features two Android activities; SplashActivity.java and MainActivity.java. The SplashActivity is the first screen shown when the app is launched. It waits for 5000ms or 5 secs and makes way for the next screen, MainActivity to appear. However, touching the SplashActivity will dismiss it immediately for the MainActivity to show.

This pattern of touch-to-dismiss is used when no resource initialization is being done during application start up. If all your splash screen does it to show off your design skills (which is perfectly fine), you've to provide a means for busy users to dismiss it immediately and get to the meat of your application.

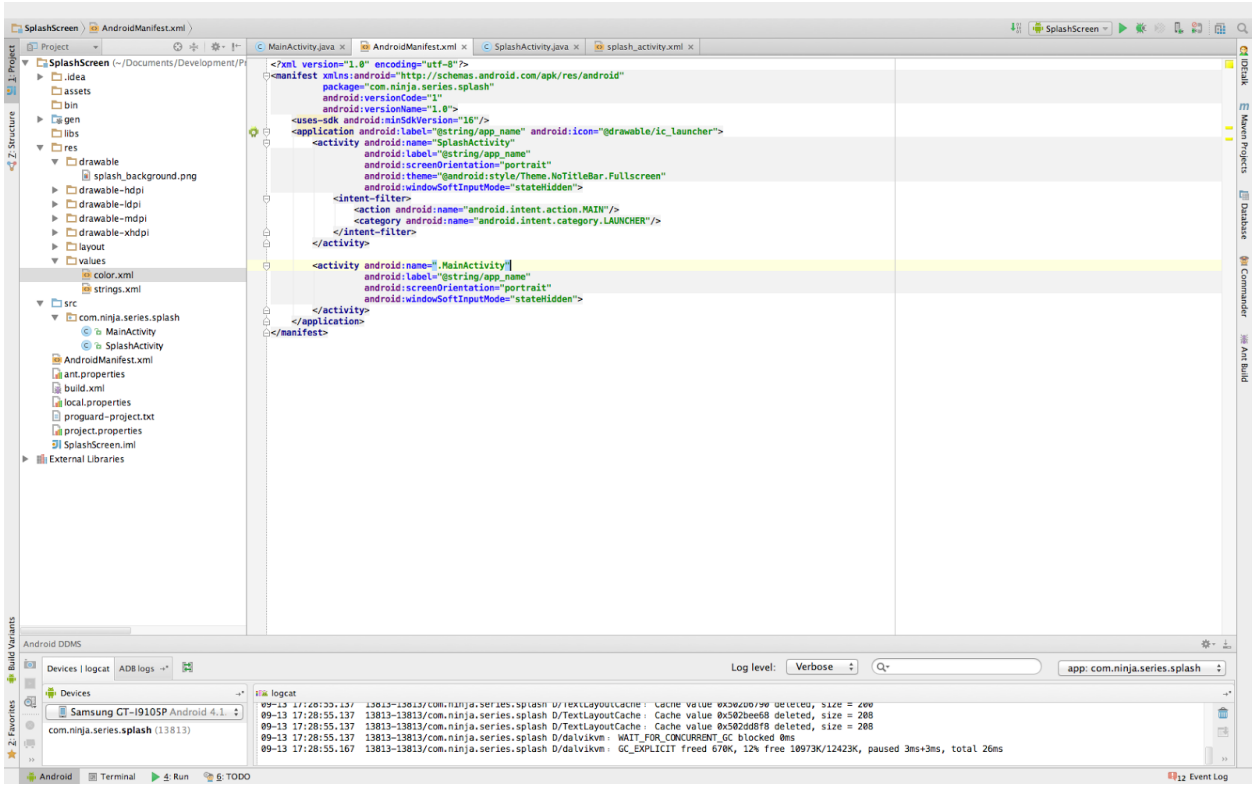
The screenshots on the next page shows the two activities used in this example project.

This is the awesome splash screen we're going to build. It is the first screen to be shown when the app is launched. It waits for 5 seconds and automatically dismisses itself for the next activity, the main activity to be shown.
However, if you touch this splash screen, it will immediately dismiss itself and make way for the main activity to be shown.



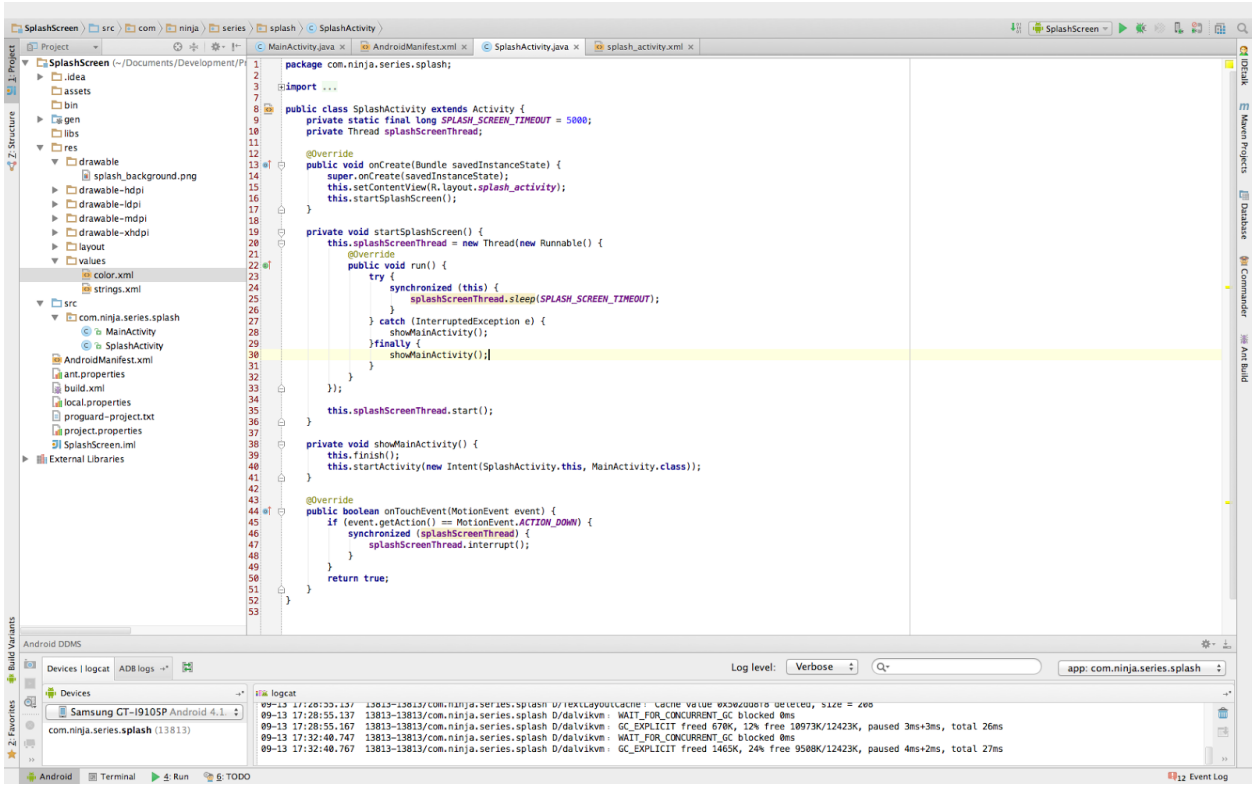
Above is the main activity. It is shown after the splash screen has been dismissed.

Configurations In The Android Manifest




```
<activity android:name="SplashActivity"
                android:label="@string/app_name"
                android:screenOrientation="portrait"
                android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
                android:windowSoftInputMode="stateHidden">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```

The highlighted text in the above xml snippet has been added to the SplashActivity in the AndroidManifest.xml to make it a fullscreen activity. Splash screens are usually shown in fullscreen mode.



The source shown in the above diagram is SplashActivity.java.

```
package com.ninja.series.splash;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.MotionEvent;

public class SplashActivity extends Activity {
    private static final long SPLASH_SCREEN_TIMEOUT = 5000;
    private Thread splashScreenThread;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.splash_activity);
        this.startSplashScreen();
    }

    private void startSplashScreen() {
        this.splashScreenThread = new Thread(new Runnable() {
            @Override
```

```
            public void run() {
                try {
                    synchronized (this) {
                        splashScreenThread.sleep(SPLASH_SCREEN_TIMEOUT);
                    }
                } catch (InterruptedException e) {
                    showMainActivity();
                }finally {
                    showMainActivity();
                }
            }
        });

        this.splashScreenThread.start();
    }

    private void showMainActivity() {
        this.finish();
        this.startActivity(new Intent(SplashActivity.this, MainActivity.class));
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            synchronized (splashScreenThread) {
                splashScreenThread.interrupt();
            }
        }
        return true;
    }
}
```

The function, startSplashScreen() defines and starts a thread which pauses for 5000 milliseconds or 5 seconds after which it shows the main activity. However, when this thread is interrupted by touching anywhere on the splash activity, it immediately dismisses the splash activity and shows the main activity.

Pay attention to the onTouchEvent in the SplashActivity.java file. This is what gets called when a user touches down on the SplashActivity. Consequently, it interrupts the splashScreenThread by calling splashScreenThread.interrupt() which results in the catch clause in the startSplashScreen method being called. In this catch clause, a call is made to the function showMainActivity which dismisses the splash screen and showns the main activity.

Thanks for reading this text. If you've any questions contact me via any of my contact links above.