Лекция #31. SplashScreen

В прошлом в Android не рекомендовалось делать Splash Screen в приложениях. Не было смысла заставлять пользователя ждать n секунд, чтобы показать заставку. Но сейчас приложения стали больше и стали дольше запускаться. Поэтому появилась необходимость в так называемых **SplashScreen** – экранов, которые замещают интерфейс приложения на время его загрузки.

В этой лекции мы рассмотрим четыре основных способа реализации Splash Screen на Android:

1. Используя Launcher Theme (**Хорошо**)
2. Используя Launcher Theme с предопределенной для Splash Screen Activity (**Сойдет**)
3. Используя таймеры (Timers) (**Плохо**)
4. Используя умные таймеры (Smart Timers) (**Ужасно**)

Используя Launcher Theme

Когда ваше приложение запускается и оно еще не в памяти, может иметь место задержка между тем, когда пользователь нажал на запуск, и тем, когда у Activity вызвано **onCreate()**. Этот, так называемый "**холодный старт**" — лучшее время, чтобы показать ваш Splash Screen.

Во время "**холодного старта**" оконный менеджер пытается отрисовать placeholder UI, используя элементы из темы приложения (app theme), такие как windowBackground. И то, что показывает **windowBackground** по-умолчанию (обычно белый или черный фон), вы можете поменять на какой угодно **drawable**, создав тем самым свой **Splash Screen**. Этот способ показывает Splash Screen только там, где необходимо, и не замедляет пользователя.

Итак, нам необходимо создать кастомную тему, переопределив **android:windowBackground**, заменив использование стандартной темы на вашу перед вызовом **super.onCreate** в нашей Activity.

Создадим файл **launch_screen.xml** в папке **res/drawable** и заполним его содержимым:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:opacity="opaque">
    <item
        android:drawable="@color/cardview_dark_background"/>
    <item
        android:drawable="@drawable/ic_launcher_foreground"
        android:gravity="center"/>
</layer-list>
```

В итоге справа в предпросмотре ресурса будет отображено следующее:

Это и будет наш ресурс для фона.

Вы должны создать новую тему AppTheme.Launcher. Единственный элемент, который необходимо переопределить — это windowBackground. В файл **res/styles.xml** добавим:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Theme.LauncherThemeSplash.Launcher">
        <item name="android:windowBackground">@drawable/launch_screen</item>
```

```
    </style>
</resources>
```

В этом примере предполагается, что главная тема приложения называется **Theme.LauncherThemeSplash**, но если это не так, просто во всех местах замените Theme.LauncherThemeSplash на имя главной темы вашего приложения.

При этом мы наследуем все остальные атрибуты главной темы **Theme.LauncherThemeSplash**, используя ее название, как префикс для названия нашей темы **Launcher**.

Пропишем тему для **Splash Screen** в файле манифеста в вашей стартовой **Activity**:

```
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:theme="@style/Theme.LauncherThemeSplash.Launcher">
```

Теперь нужно вернуть главную тему в стартовую Activity (если, конечно, мы не хотим, чтобы Splash Screen радовал нас и во время работы приложения)

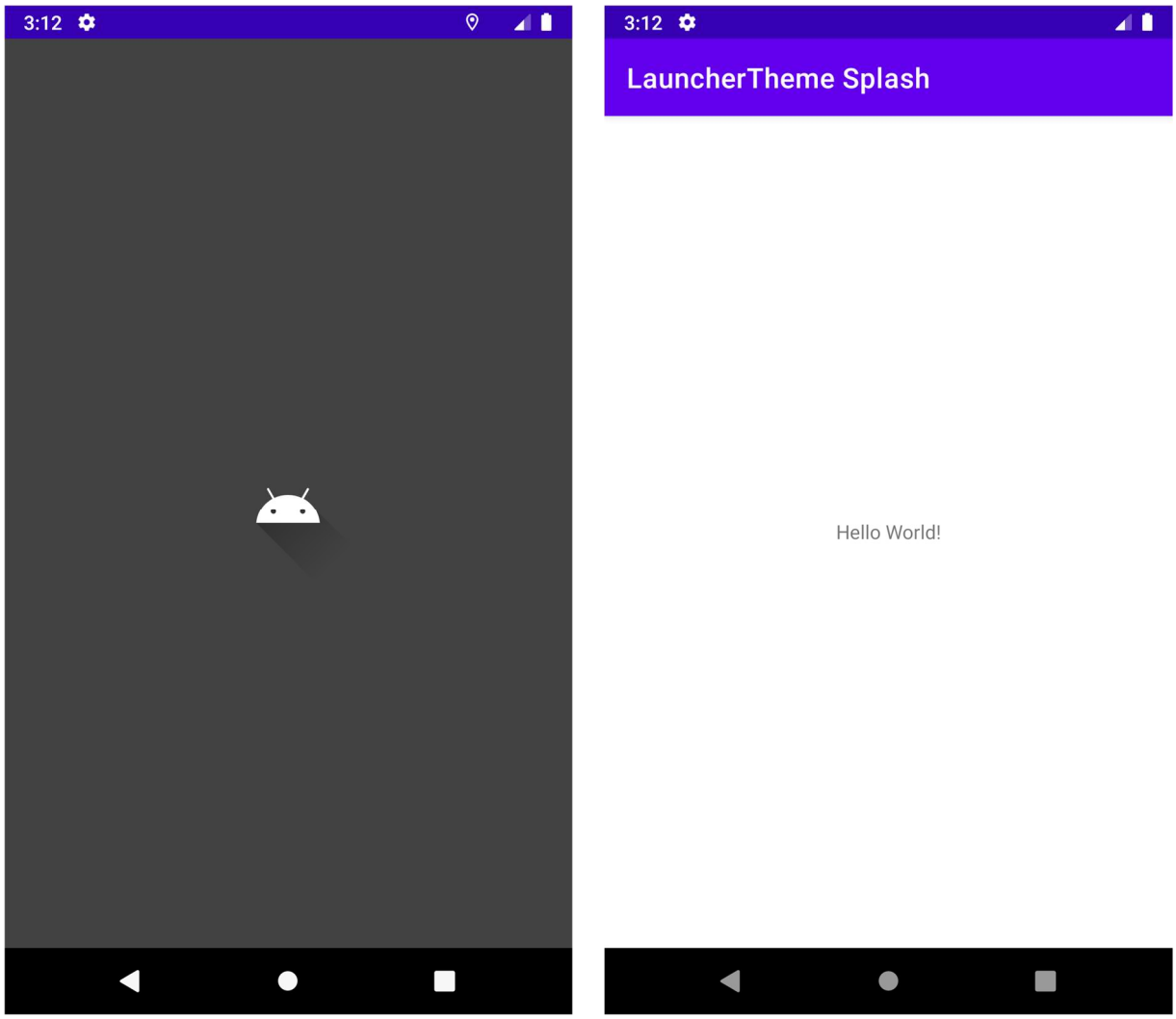
Самый простой способ сделать это — это вызвать **setTheme(R.style.AppTheme)** до super.**onCreate()** и **setContentView()**:

```
package com.awkitsune.launcherthemesplash

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        setTheme(R.style.Theme_LauncherThemeSplash)

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```



Splash Screen в отдельной Activity с использованием Launcher Theme и таймером

Этот способ базируется на первом способе. Он требует отдельной **Activity** для Splash Screen. Первые два шага пропускаем, они аналогичны первому способу.

Осталось создать **Activity** для **Splash Screen** и указать в манифесте для нее тему **Theme.LauncherThemeSplash.Launcher**:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.awkitsune.launcherthemesplash">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.LauncherThemeSplash">
        <activity
            android:name=".SplashActivity"
            android:exported="true"
```

```
            android:theme="@style/Theme.AppCompat.NoActionBar"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="true">
        </activity>
    </application>

</manifest>
```

Так же теперь в файле разметки укажем фон **SplashActivity**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SplashActivity"
    android:background="@drawable/launch_screen">

</androidx.constraintlayout.widget.ConstraintLayout>
```

Теперь отредактируем **SplashActivity** так, чтобы она перенаправляла на **MainActivity**:

```
package com.awkitsune.launcherthemesplash

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.os.Looper

class SplashActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

        scheduleSplashScreen()
    }

    private fun scheduleSplashScreen() {
        val splashScreenDuration: Long = 2000

        Handler(Looper.getMainLooper()).postDelayed({
            val intent = Intent(this@SplashActivity,
MainActivity::class.java)
            startActivity(intent)

            finish()
        }, splashScreenDuration)
    }
}
```

И очистим код **MainActivity**:

```
package com.awkitsune.launcherthemesplash

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

В итоге мы получим результат, похожий на предыдущий, но с более гибкой возможностью настройки анимаций, дизайна, действий