**Создание анимации для Splash экрана**

В разделе [Анимация](http://developer.alexanderklimov.ru/android/animation.php) есть статья, посвященная [анимации преобразований](http://developer.alexanderklimov.ru/android/tweenanimation.php). Здесь мы лишь закрепим изученный материал. В данном примере использовались материалы из книги [Android за 24 часа. Программирование приложений под операционную систему Google](http://www.ozon.ru/context/detail/id/6728228/?partner=visual), которую рекомендую купить.

Ссылка на исходный код приведена в конце статьи

**Разработка дизайна экрана-заставки**

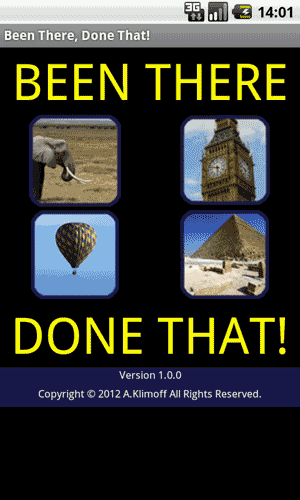
Одним из примеров, где можно использовать анимационные эффекты - это экран-заставка игры, которая появляется на несколько секунд, а затем исчезает, уступая место другому экрану-меню.

Если на таком экране-заставке вывести просто текст о названии и версии игры, а также логотип, то будет скучно. Мы пойдем по другому пути. Сделаем следующее.

* Поместим на форму TextView для отображения верхней текстовой строки
* Далее разместим контейнер TableLayout с двумя строками и столбцами - получается четыре ячейки
* В каждой ячейке разместим элемент ImageView
* Снова размещаем TextView с другой текстовой информацией
* И снова размещаем TextView, где будет содержаться информация о версии программы

Далее добавляем ресурсы в проект: картинки (res/drawable), цвета для текстовой информации (colors.xml), размеры для шрифтов (dimens.xml).

В итоге мы получим следующий результат



**Создание анимации**

Будем использовать анимацию преобразований. Создаем папку **res/anim**, в котором создадим три новых файла **fade\_in.xml**, **fade\_in2.xml** и **custom\_anim.xml**.

В первом файле анимация плавно увеличивает непрозрачность (альфа-канал) верхнего текста от значения 0 (прозрачный) до значения 1 (непрозрачный) в течение 2.5 секунд.

Во втором файле происходит тоже самое, только добавлен еще один атрибут **startOffset**. Это означает, что сначала будет задержка в течение 2.5 секунд, а только потом произойдёт анимация с такой же продолжительностью. Её мы применим к нижнему тексту. Получается, что сначала произойдет анимация верхнего текста, а потом анимация нижнего текста.

Третья анимация самая интересная. Она будет вращать картинки вокруг своей оси (rotate), а также менять их прозрачность (alpha). Причём вращаться они будут парами. Сначала будут вращаться левые две картинки, а затем - правые. Кроме вращения и изменения прозрачности, также будут меняться их размеры (scale).

Управление анимации происходит программным путём. Чтобы применить первую анимацию к верхнему тексту, нужно получить экземпляр элемента **TextView** в методе **onCreate()**, загрузить анимационный ресурс в объект **Animation** и вызвать метод **startAnimation()** элемента **TextView**.

TextView logo1 = (TextView) findViewById(R.id.TextViewTopTitle);

Animation fade1 = AnimationUtils.loadAnimation(this, R.anim.fade\_in);

logo1.startAnimation(fade1);

Аналогично поступаем и со вторым текстовым элементом.

Чтобы остановить анимацию, например, в методе **onPause()**, нужно вызвать метод **clearAnimation()**.

**Анимация всех элементов в элементе-контейнере**

Помимо возможности применения анимации к отдельным элементам, вы можете применить анимацию к каждому элементу **View**, содержащему в элементе-контейнере (например, **TableLayout** и к каждому элементу **TableRow**), используя класс **LayoutAnimationController**.

В этом случае нужно загрузить соответствующую анимацию, создать экземпляр класса **LayoutAnimationCotroller**, произвести необходимые настройки этого экземпляра и затем вызвать метод **setLayoutAnimation()** элемента-контейнера.

Animation spinin = AnimationUtils.loadAnimation(this, R.anim.custom\_anim);

LayoutAnimationController controller = new LayoutAnimationController(spinin);

TableLayout table = (TableLayout)findViewById(R.id.TableLayout01);

for (int i = 0; i < table.getChildCount(); i++){

TableRow row = (TableRow)table.getChildAt(i);

row.setLayoutAnimation(controller);

}

В данном случае нет необходимости вызывать метод **startAnimation()**, поскольку это сделает экземпляр класса **LayoutAnimationController**. При использовании данного подхода анимация применяется к каждому дочернему элементу, однако воспроизведение каждой анимации начинается в разное время. Это создает красивый эффект, когда каждый элемент поочередно делает оборот вокруг своего центра.

Если после второй строчки вставить метод **setOrder()**, то анимация пар картинок будет происходить в случайном порядке:

controller.setOrder(2);

[Скачать исходный код](http://developer.alexanderklimov.ru/android/apk/Chapter7code.zip)