

### Лабораторная работа №6. Работа с анимацией

Задание: создать приложение, содержащее анимированные интерфейсные элементы (например, увеличивающиеся при клике на них кнопки, вращающиеся TextView и т.д.).

Самым простым видом анимации является покадровая анимация, когда картинки (кадры) сменяют друг друга, создавая эффект движения или изменения изображения. Это слишком простой, но утомительный процесс, поэтому рассмотрим анимацию другого типа, так называемую tween-анимацию. Создайте новое приложение с Empty Activity, перейдите в файл activity\_main.xml, удалите текстовое поле и поместите в центр экрана кнопку, а внизу – как обычно, текстовое поле с ФИО. Затем необходимо создать новый xml-файл для описания анимации. Для этого нажмите правой кнопкой мыши по папке res и создайте новый файл Android resource с параметрами, которые изображены на рисунке 27.

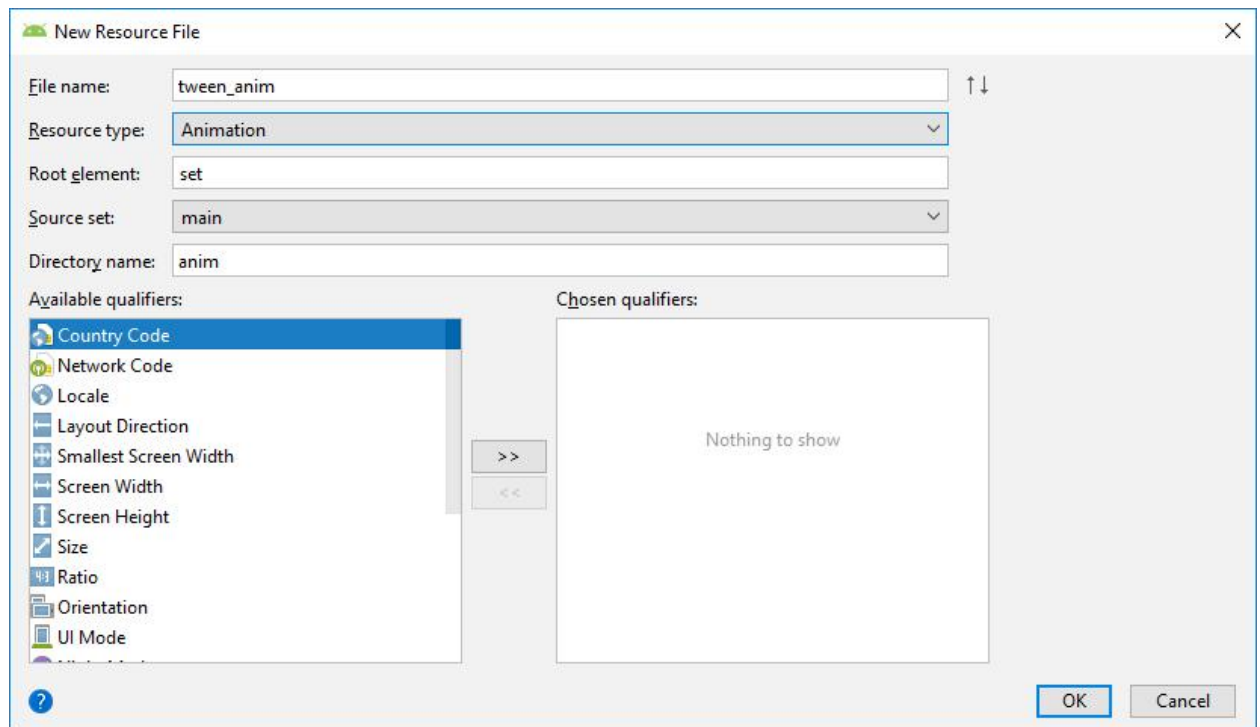


Рисунок 27 – Создание файла анимации

После данных действий в папке res появится новая папка anim с файлом tween\_anim.xml внутри. Переходим в этот файл и добавляем следующий код внутри тега set:

```
<scale
    android:fromXScale="2.0"
    android:toXScale="0.5"
    android:fromYScale="2.0"
    android:toYScale="0.5"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="2000" />
```

Здесь описывается уменьшение объекта с длительность анимации в 2 секунды (**android:duration**). Так как анимация будет отображаться на кнопке, нужно будет вернуть её в исходный размер, для этого добавляем ещё немного кода:

```
<scale
  android:fromXScale="0.5"
  android:toXScale="2.0"
  android:fromYScale="0.5"
  android:toYScale="2.0"
  android:pivotX="50%"
  android:pivotY="50%"
  android:duration="2000"
  android:startOffset="2000" />
```

Теперь кнопка сможет обратно увеличиться до своих размеров, анимация увеличения также будет длиться 2 секунды, но она будет отработывать не сразу, а с задержкой в 2 секунды (**android:startOffset**). То есть сначала идёт анимация уменьшения объекта, а затем анимация увеличения. Ну и в заключение добавим анимацию вращения кнопки:

```
<rotate
  android:pivotX="50%"
  android:pivotY="50%"
  android:fromDegrees="0"
  android:toDegrees="360"
  android:duration="2000"
  android:startOffset="4000" />
```

Анимация вращения будет ждать 4 секунды (пока отработают первые две анимации), а затем за 2 секунды провернёт кнопку на 360 градусов по часовой стрелке.

Ещё одним важным свойством анимации является параметр **interpolator**. Интерполятор позволяет ускорить или замедлить выполнение анимации на устройстве, а также добавить некоторые эффекты. Добавим следующий код в конец открывающего тэга **set**:

```
android:interpolator="@android:anim/bounce_interpolator"
```

Набрав **@android:anim** в кавычках, можно увидеть, какие типы интерполяторов бывают, см. рисунок 28. Можно поэкспериментировать с разными типами интерполяторов.

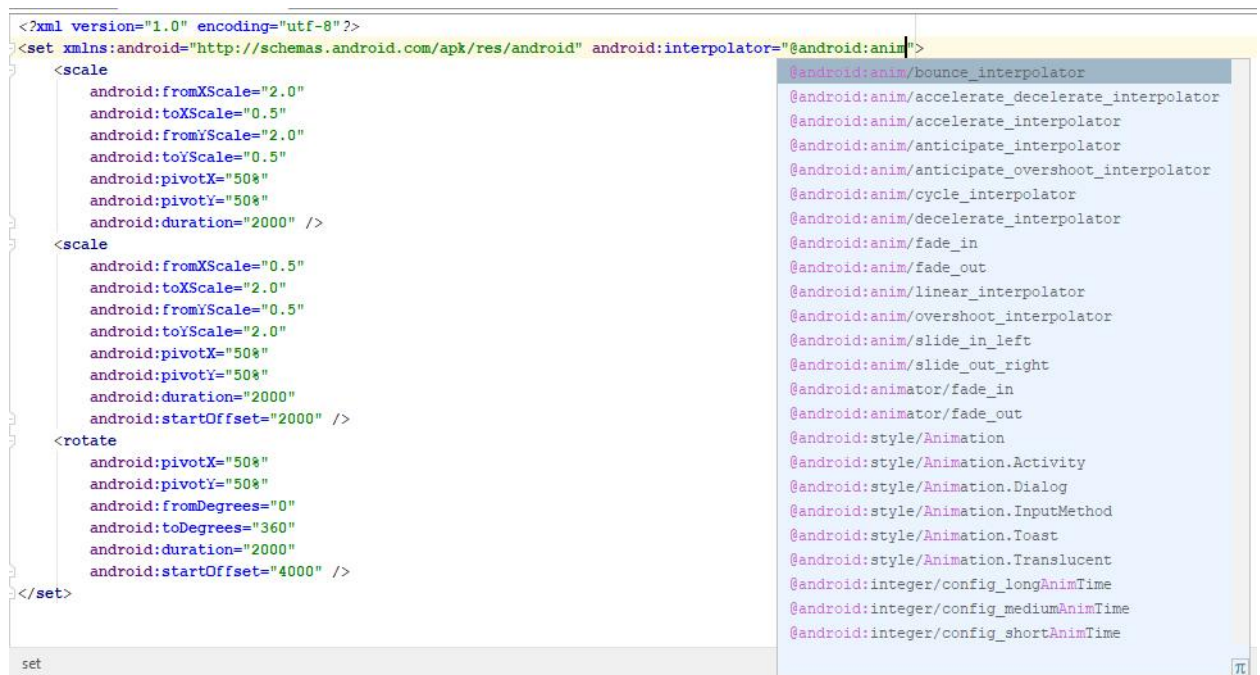


Рисунок 28 – Варианты интерполяторов

Итак, в нашем случае открывающий тэг set должен выглядеть следующим образом:

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/bounce_interpolator">
```

С описанием анимации закончили, теперь необходимо перейти в файл MainActivity.java и выполнить анимацию после нажатия на кнопку. Для этого в методе onCreate создаём ссылку на кнопку:

```
final Button button = findViewById(R.id.button);
```

И создаём для неё listener и onClick. В onClick добавляем анимацию:

```
Animation animation = AnimationUtils.loadAnimation(MainActivity.this, R.anim.tween_anim);
button.startAnimation(animation);
```

Импортируем все зависимости и исправляем ошибки, после чего запускаем проект на эмуляторе. Теперь необходимо изменить параметры в файле `tween_anim.xml` на собственные, чтобы все не сдавали одну и ту же анимацию. Поэкспериментируйте с различными параметрами, можете также производить анимацию не с кнопкой, а с каким-нибудь другим элементом. Результат выполнения работы показан на рисунке 29.

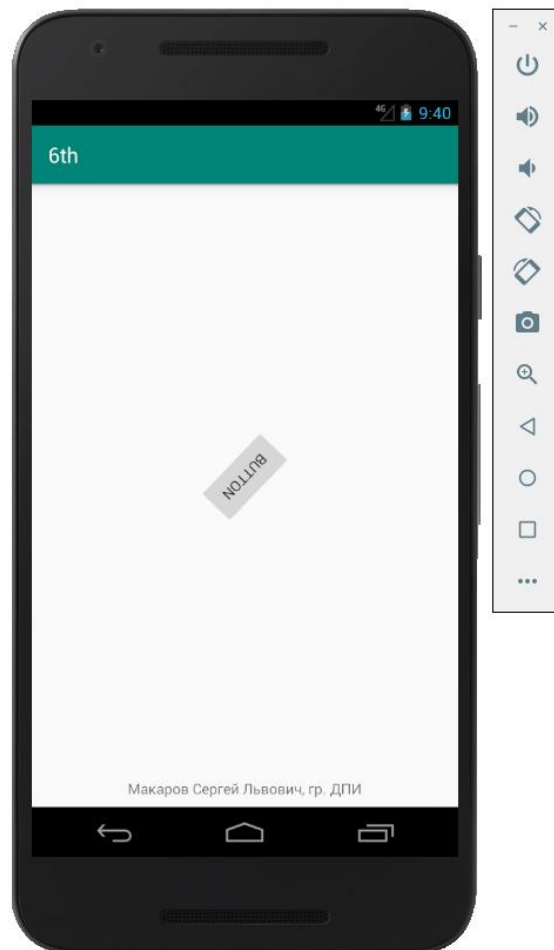


Рисунок 29 – Результат анимации