Лабораторная работа 7. Разработка интерфейсов мобильных приложений

Цели лабораторной работы:

- Изучить основы разработки интерфейсов мобильных приложений
- Научиться создавать приложения, состоящие из нескольких активностей, и диалоговые окна, а также познакомиться с элементами тач-интерфейса

Задачи лабораторной работы:

- Изучить элементы интерфейса.
- Изучить макеты (layouts).
- Практическим путём научиться размещать элементы и менять их свойства.
- Научиться создавать многооконные приложения.
- Научиться создавать диалоговые окна и всплывающие подсказки.
- Научиться проектировать приложения со слайдингом.
- Научиться использовать анимацию.
- Доработать интерфейс приложений «Угадай число» и «Калькулятор».

Table of Contents

Критерии оценивания

4-5 — приложения на основе примеров 1-5 из лабораторной работы и ответы на контрольные вопросы;

6-7 — приложения на основе примеров из лабораторной работы, задание №1 для самостоятельной работы и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается**;

8 — приложения на основе примеров из лабораторной работы, задания № 1-2 для самостоятельной работы на языке Java и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается**;

9 — приложения на основе примеров из лабораторной работы, все задания для самостоятельной работы, в том числе задание 2 реализовать на языке Kotlin, дополнительный функционал, предложенный студентом самостоятельно, и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается.**

Основы разработки интерфейсов мобильных приложений

Лабораторная работа посвящена разработке интерфейсов мобильных приложений. Работа содержит подробное описание построения гармоничного понятного пользовательского интерфейса для главной активности приложения и описание основных элементов интерфейса. Лабораторная работа поможет выбрать концепцию своего приложения и начать разработку его интерфейса.

Более подробную информацию о разработке интерфейсов мобильных приложений под Android можно узнать на сайте Design [Android Developers].

ОСНОВЫ ВЕРСТКИ

Цель — изучить основы верстки. Научиться управлять интерфейсом мобильного устройства при разработке программного приложения.

Просмотрите основные сведения о классах, которые понадобятся при разработке приложения.

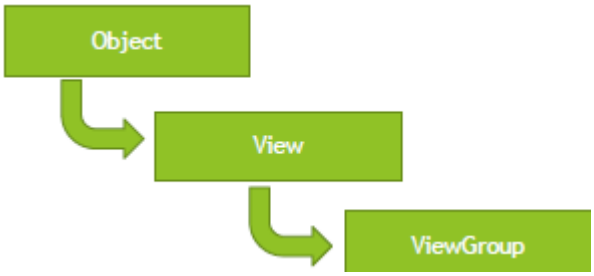

Рис. Иерархия классов View и ViewGroup

Дерево представлений для Activity представлено на рисунке ниже.

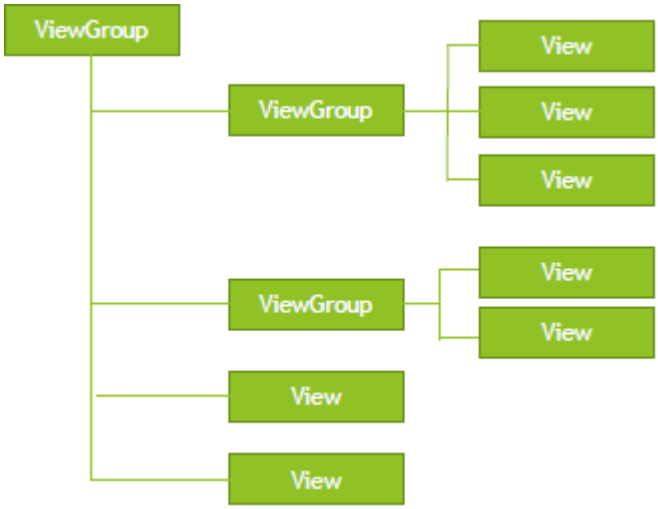

Рис. Дерево представлений для Activity

Файл разметки имеет следующую структуру

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="0.25"
android:padding="5dp">
<Button
android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button1"
android:id="@+id/button3" android:layout_gravity="right"/>
<Button
android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
```

```
android:text="Button2"
android:id="@+id/button"/>
<Button
android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button3"
android:id="@+id/button2" android:layout_gravity="center_horizontal"/>
</LinearLayout>
<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:paddingLeft="40dp"
android:paddingRight="40dp"
android:layout_weight="0.5"
android:gravity="center_vertical">
<Button
android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button4"
android:id="@+id/button3"/>
<Button
android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button5"
android:id="@+id/button"/>
</LinearLayout>
<LinearLayout
android:layout_width="match_parent"
```

android:layout_height="0dp"

android:layout_weight="0.25"

android:padding="5dp"

android:gravity="bottom">

...

</LinearLayout>

</LinearLayout>

Распространенные виды макетов

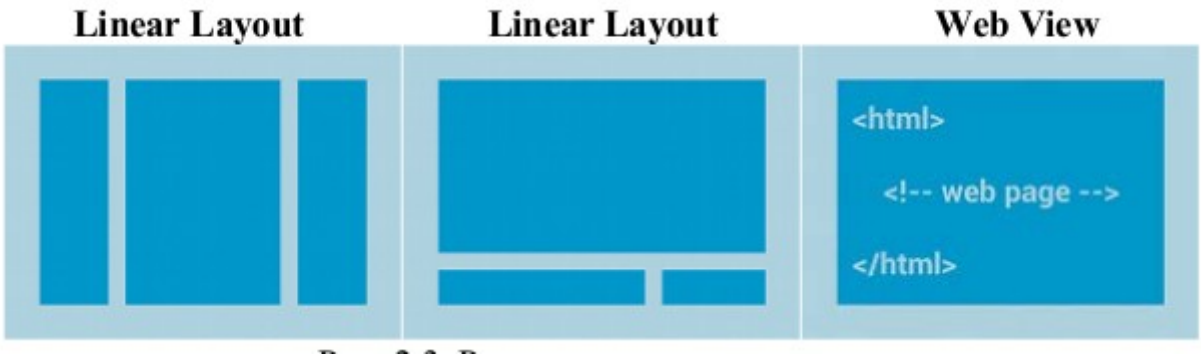

Рис. Распространенные виды макетов

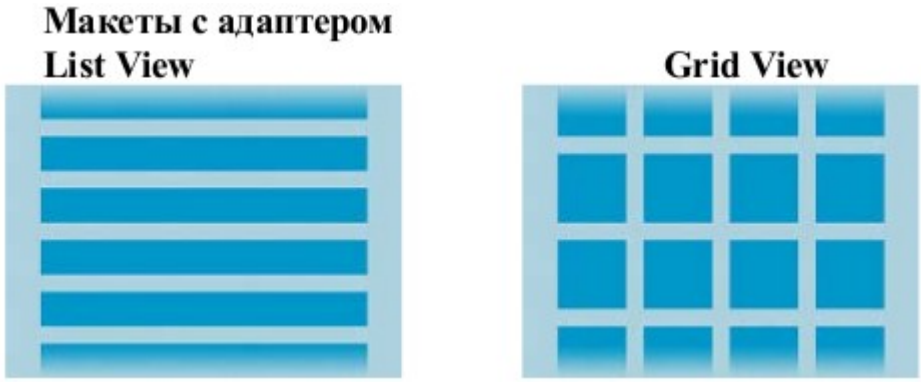

Рис. Интерфейс макетов с адаптером

Атрибуты LinearLayout

Attribute Name	*Related Method*	*Description*
android:baselineAligned	setBaselineAligned(boolean)	When set to false, prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	setBaselineAlignedChildIndex(int)	When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align to (that is, which child TextView).
android:divider	setDividerDrawable(Drawable)	Drawable to use as a vertical divider between buttons.
android:gravity	setGravity(int)	Specifies how an object should position its content, on both the X and Y axes, within its own bounds.
android:measureWithLargestChild	setMeasureWithLargestChildEnabled(boolean)	When set to true, all children with a weight will be considered having the minimum size of the largest child.
android:orientation	setOrientation(int)	Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.
android:weightSum		Defines the maximum weight sum.

УПРАЖНЕНИЕ 1. МАКЕТЫ ПРИЛОЖЕНИЙ ДЛЯ ПЛАТФОРМЫ ANDROID

Изучить материалы по основам верстки и :

https://www.coderefer.com/android-layouts-types-android-layouts/

https://developer.android.com/guide/topics/ui/declaring-layout

https://developer.android.com/guide/components/activities/intro-activities

https://developer.android.com/guide/components/fragments

Реализовать пример 1

Пример 1. Создание прототипа интерфейса

Рассмотрим пример разработки интерфейса приложения, которое ищет в сети Интернет изображения по запросу пользователя, позволяет оценивать их, скачивать, и посещать интернет-страницы сайтов, на которых было найдено изображение. В приложении используются макеты LinerLayout, FrameLayout, загружаемое изображение.

Создание заготовки для приложения

Выглядеть главное окно будет примерно так:

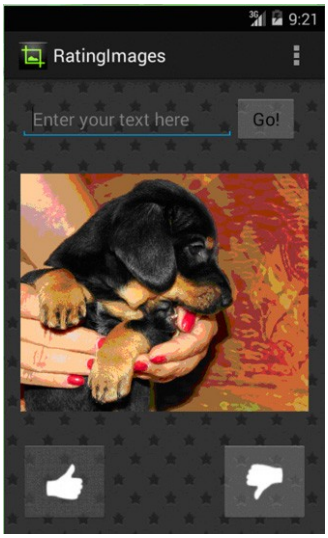

Рис. 1. Интерфейс главной активности

На нём присутствуют поле ввода текста для запроса пользователя и кнопка, начинающая поиск изображений. Внизу экрана две кнопки: "**like**" и "**dislike**", с их помощью пользователь сможет оценить изображение. После того, как пользователь сделает оценку изображения, текущее изображение закрывается и загружается следующее.

Итак, начнём с создания нового проекта. Назовём его "**RatingImages**" ("Рейтинг изображений").

На данном этапе изучения программирования под Android не обязательно менять иконку запуска, но эта лабораторная работа направлена на разработку дизайна интерфейса, и потому, приступим.

Создайте иконку на свой вкус.

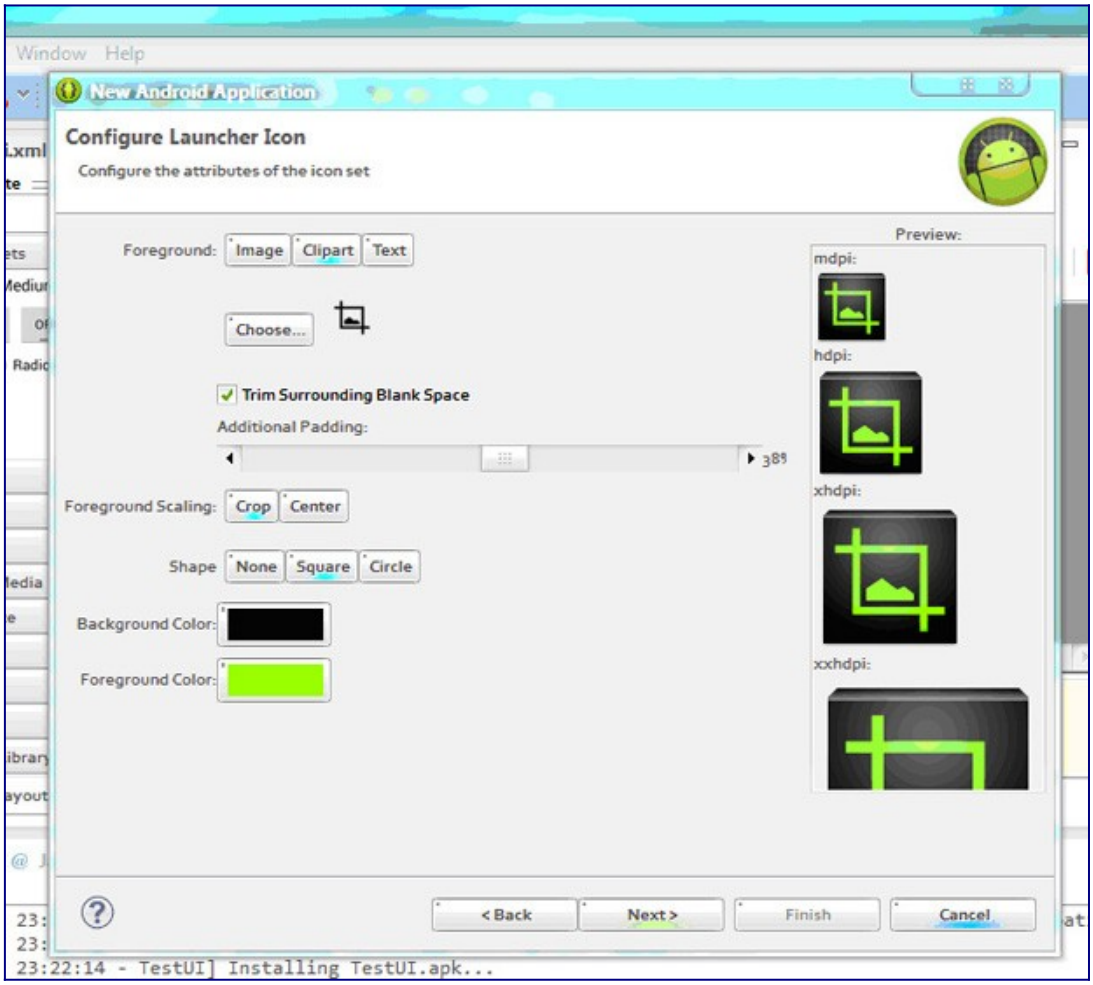

Рис. 2. Создание иконки

После создания проекта откройте **activity_main.xml** из каталога **res/layout/**.

Когда вы откроете файл **activity_main.xml**, вы увидите графический редактор макета. Благодаря этому редактору создание интерфейсов стало ещё интереснее, поскольку добавить элемент на форму можно при помощи перетаскивания мышью, к тому же, благодаря графическому редактору, не обязательно запускать эмулятор, чтобы увидеть результат своих трудов.

Теперь щелкните по вкладке **activity_main.xml** в нижней части экрана. Открылся XML-редактор кода. Этот способ редактирования стандартный, но все изменения, вносимые в этот документ, можно так же ощутить визуально, перейдя на графический редактор.

Вернёмся на вкладку с графическим редактором. Во-первых, подготовим документ к началу работы, для этого удалите <TextView>.

Результат выглядит так:

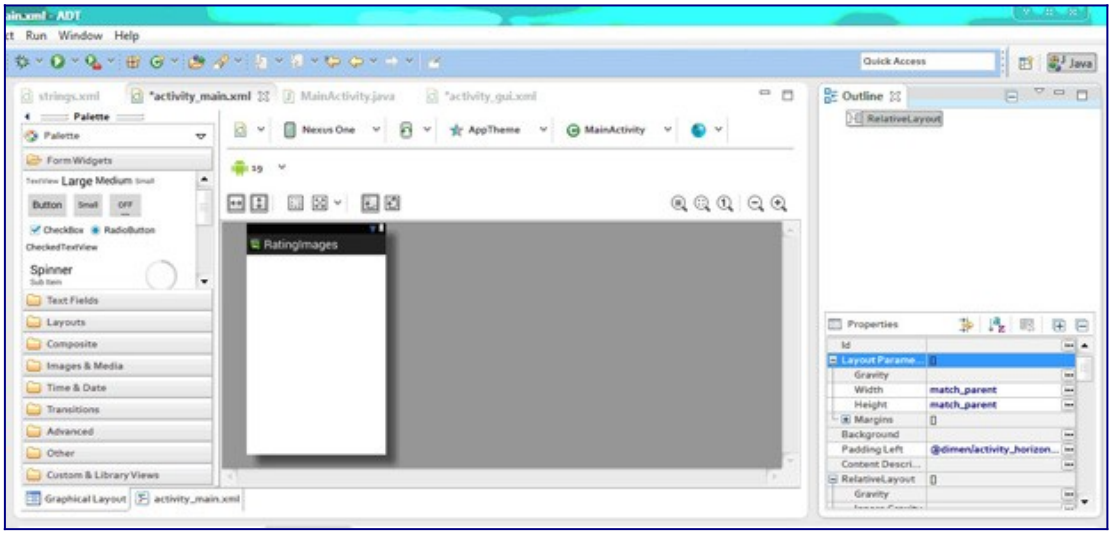

Рис. 3. Проект, готовый к началу разработки

На рабочей области экрана остался один элемент. Это макет <RelativeLayout>. В нём позиция дочерних элементов может быть описана по отношению друг к другу или к родителю. Подробнее о макетах можно узнать здесь.

Два атрибута, ширина и высота (android:layout_width и android:layout_height), требуются для всех элементов для того, чтобы указать их размер.

Так как <RelativeLayout> - это корень в макете, то нужно, чтобы он заполнял всю область экрана. Это достигается при помощи установки параметра "match_parent" для ширины и высоты. Это значение указывает, что ширина и высота элемента будет равна ширине и высоте родителя.

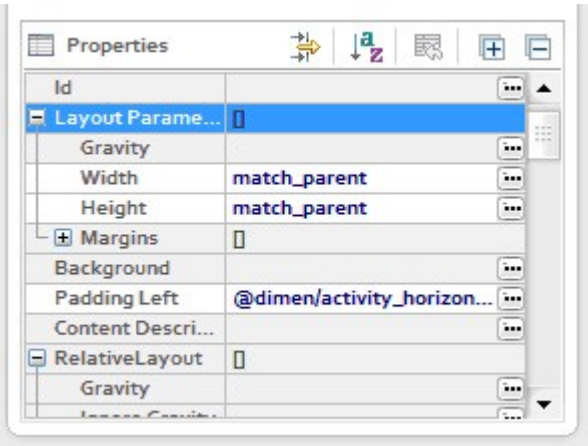

Рис. 4. Свойства <RelativeLayout> элемента

Как установить это значение?

Однократным щелчком левой кнопкой мыши по надписи "**Width**" активируйте строку с параметрами ширины:

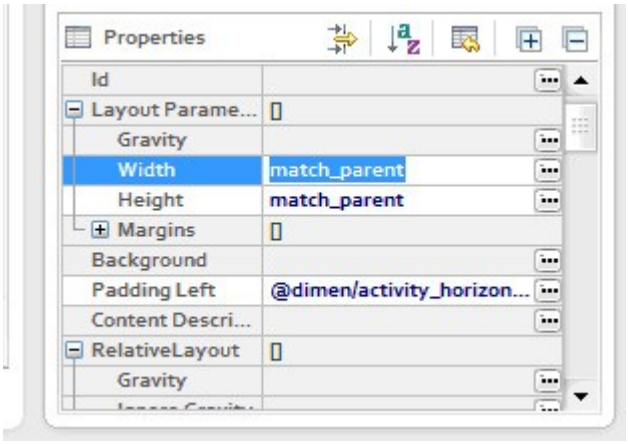

Рис. 5. Свойства <RelativeLayout> элемента, выбран атрибут android:layout_width

Щелчком левой кнопки мыши по области ввода вызовите диалоговое окно, и двойным щелчком сделайте выбор параметра:

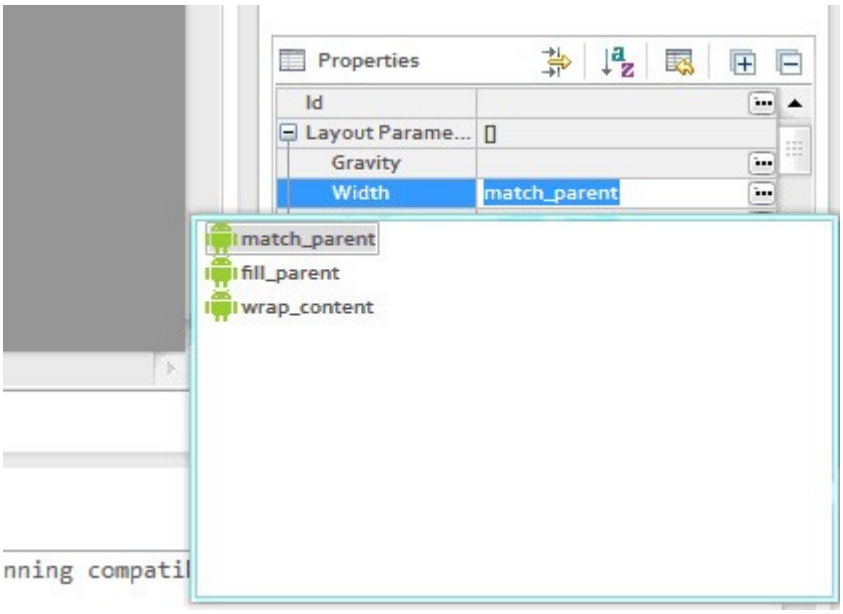

Рис. 6. Свойства <RelativeLayout> элемента, выбран параметр "match_parent"

Или щелчком правой кнопки мыши по <RelativeLayout> в Outline:

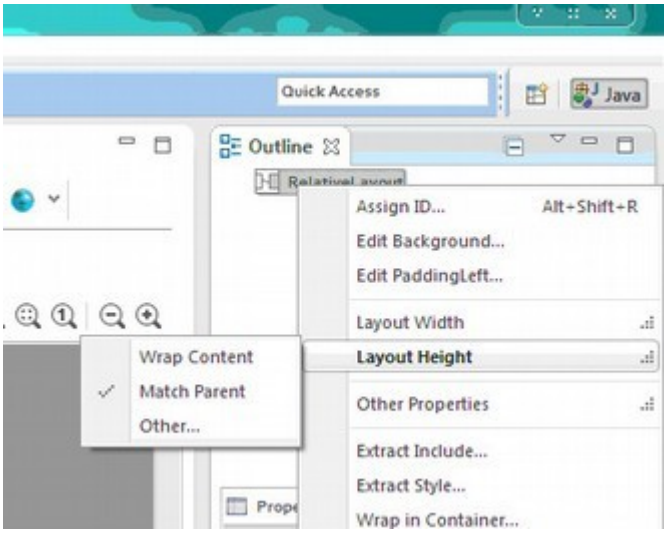

Рис. 7. Контекстное меню <RelativeLayout> элемента

При выполнении первого способа вы увидели еще два возможных параметра: "fill_parent" и "wrap_content".

На самом деле, match_parent = fill_parent, но "fill_parent" считается устаревшим, и к использованию в новых проектах предлагается "match_parent".

Параметр "wrap_content" указывает, что представление будет увеличиваться при необходимости, чтобы поддерживать соответствие содержанию экрана.

Добавление текстового поля

Для начала добавьте элемент <LinearLayout> с горизонтальной ориентацией в <RelativeLayout>, и укажите для ширины и высоты параметр "wrap_content". Теперь, для создания пользовательского редактируемого текстового поля, добавьте элемент <EditText> с параметром "wrap_content" для ширины и высоты в <LinearLayout>.

Сейчас должно получиться примерно следующее:

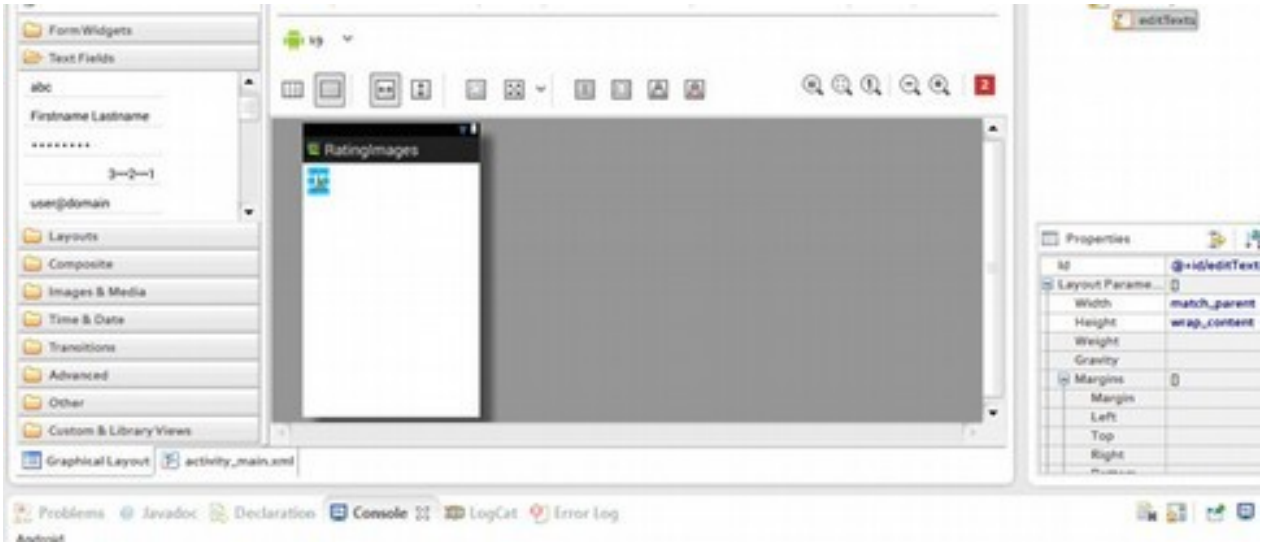

Рис. 8. Добавление текстового поля

Возможно, появился желтый предупреждающий знак, но сейчас это не важно, со временем он исчезнет. Наличие таких предупреждений никак не влияет на компилируемость проекта.

Теперь переходим к настройке добавленных нами элементов.

Для многих элементов нужно назначать id, он обеспечивает уникальный идентификатор, который можно использовать как ссылку на объект из кода вашего приложения для управления им.

Откроем редактор кода XML-файла и обратим внимание на элемент

```
<EditText>:
        <EditText
                android:id="@+id/editText1"
                android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content">
            <requestFocus/>
    </EditText>
```

Строка <requestFocus/> появляется добавлением элемента requestFocus (папка Advanced), и позволяет установить фокус на нужном компоненте. Важно использовать этот элемент, когда у вас имеется, к примеру, три текстовых поля, и нужно, чтобы фокус был на втором из них.

При указании id, знак (@) требуется в том случае, если вы имеете в виду любой ресурс объекта из XML-файла. За ним следуют тип ресурса (в данном случае ID), косая черта (слеш) и имя ресурса (editText1).

Знак плюс (+) перед типом ресурсов необходим только тогда, когда вы впервые определяете идентификатор ресурса.

По сути, id, который создается автоматически, уже уникален, но грамотнее переименовывать id в соответствии со назначением элемента.

Зададим id для текстового поля. Для этого прямо в коде строку android:id="@+id/editText1" заменяем на android:id="@+id/edit_message", жмём **CTRL+S** и открываем графический редактор. Если всё хорошо, то в свойствах текстового поля в графе **id** будет следующее:

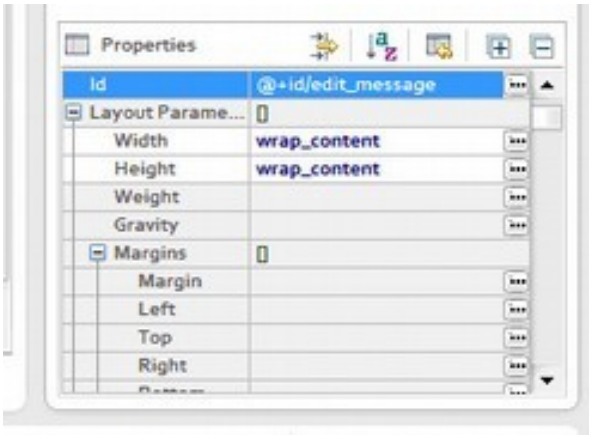

Рис. 9. Идентификатор текстового поля

Добавим в код ещё две строки:

android:ems="10" - задает соответствия для симметричного отображения шрифтов,

android:hint="@string/edit_message" - содержание тестового поля "по умолчанию", т.е. пока пользователь не начал вводить в поле текст. Вместо того, чтобы использовать просто слово (например android:hint="message"), что крайне не удобно при изменении основного языка приложения, используется ссылка на значение, хранящееся в файле **strings.xml**. Поскольку это относится к конкретному

ресурсу (а не только к **id**), знак плюс не нужен.

Однако, мы ещё не определили строку ресурсов файле **strings.xml**, и потому вы получите следующее:

Рис. 10. Предупреждение: не найден ресурс, на который прописана ссылка

Для того, чтобы ссылка на ресурс начала работать, нужно этот ресурс создать.

Откройте файл **res/values/strings.xml**. Очевидно, что его тоже можно редактировать двумя способами: графически и вручную. Выберите любой из способов.

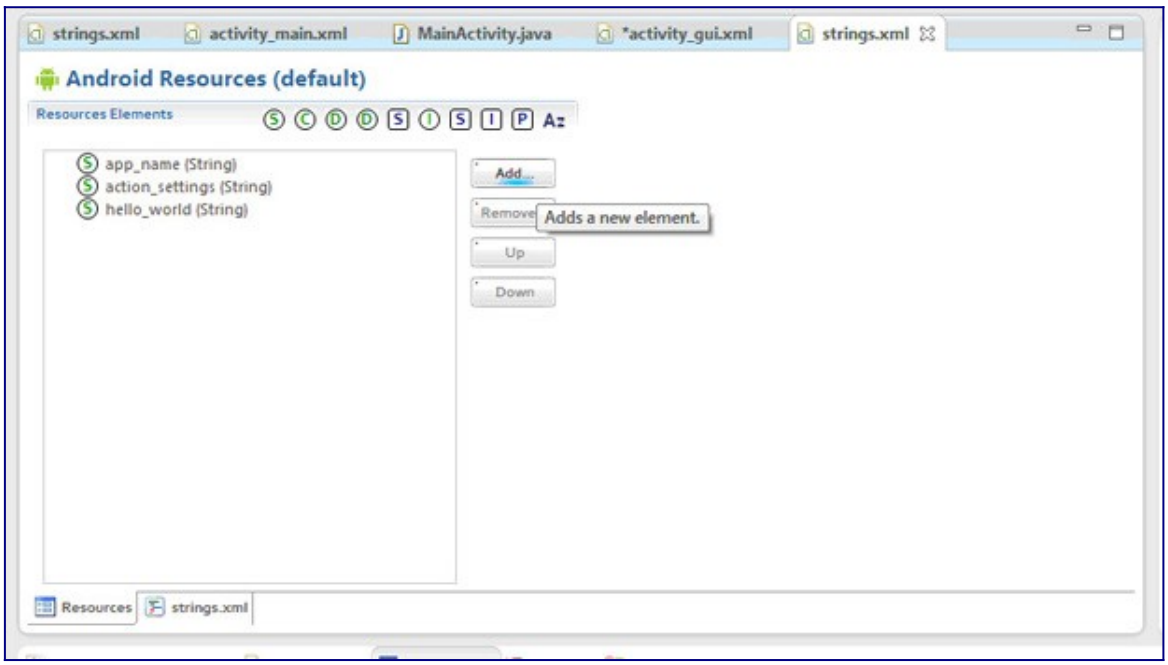

Рис. 11. Редактирование файла ресурсов графическим способом; добавление нового ресурса

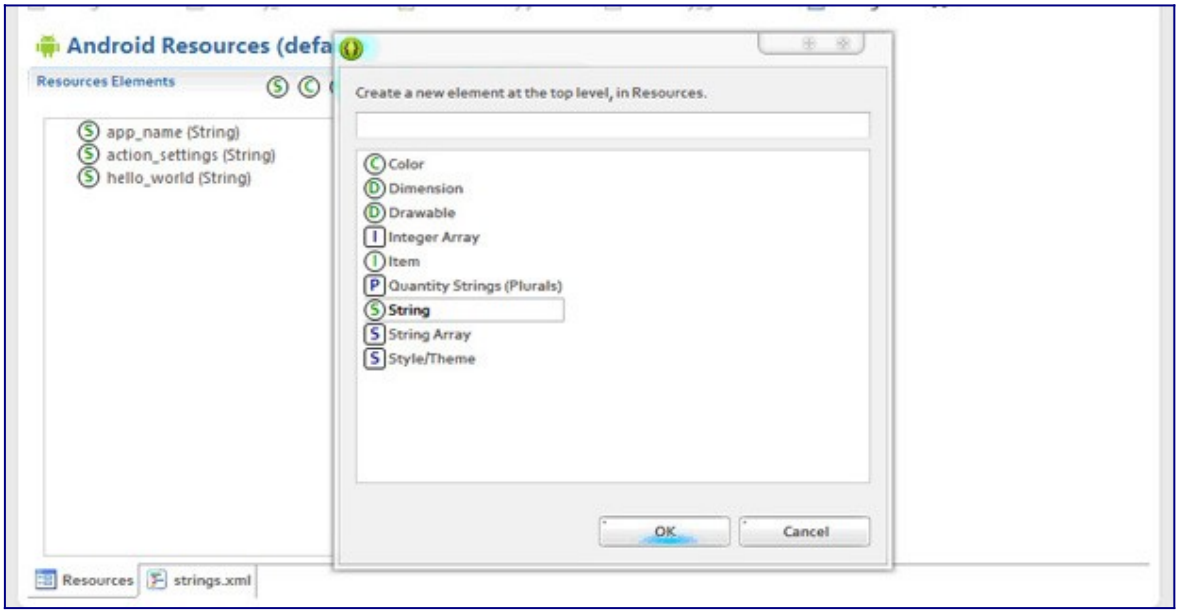

Рис. 12. Редактирование файла ресурсов графическим способом; выбор типа ресурса, выбран тип String

Заполняем поля "**Имя**" и "**Значение**":

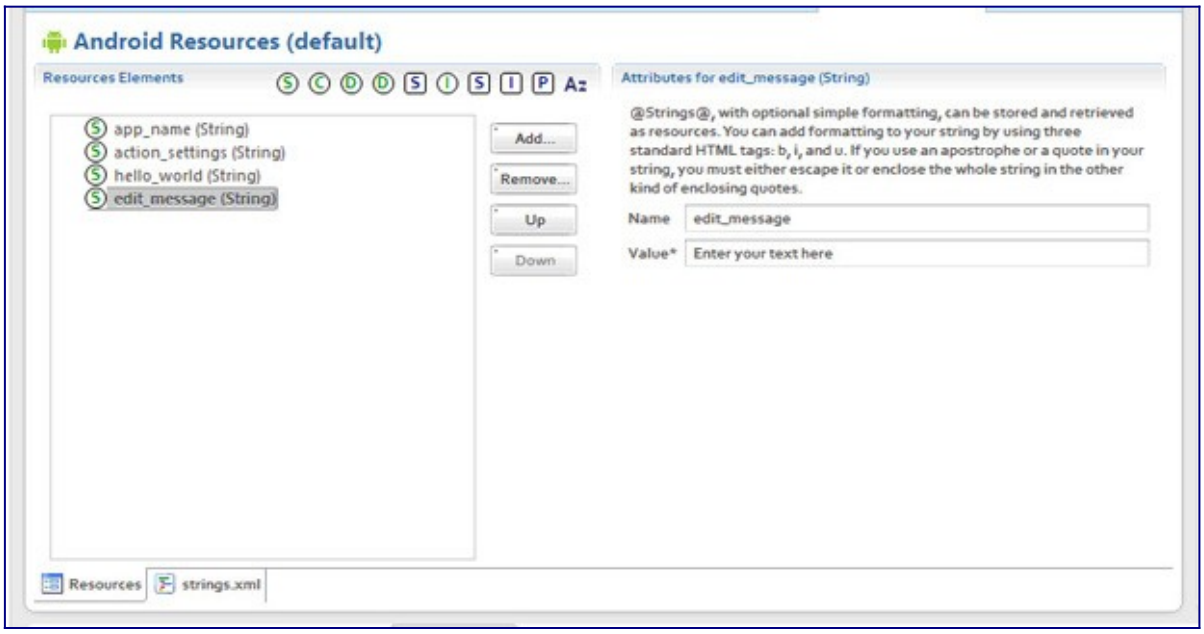

Рис..13. Редактирование файла ресурсов графическим способом

Сохраняем и любуемся результатом:

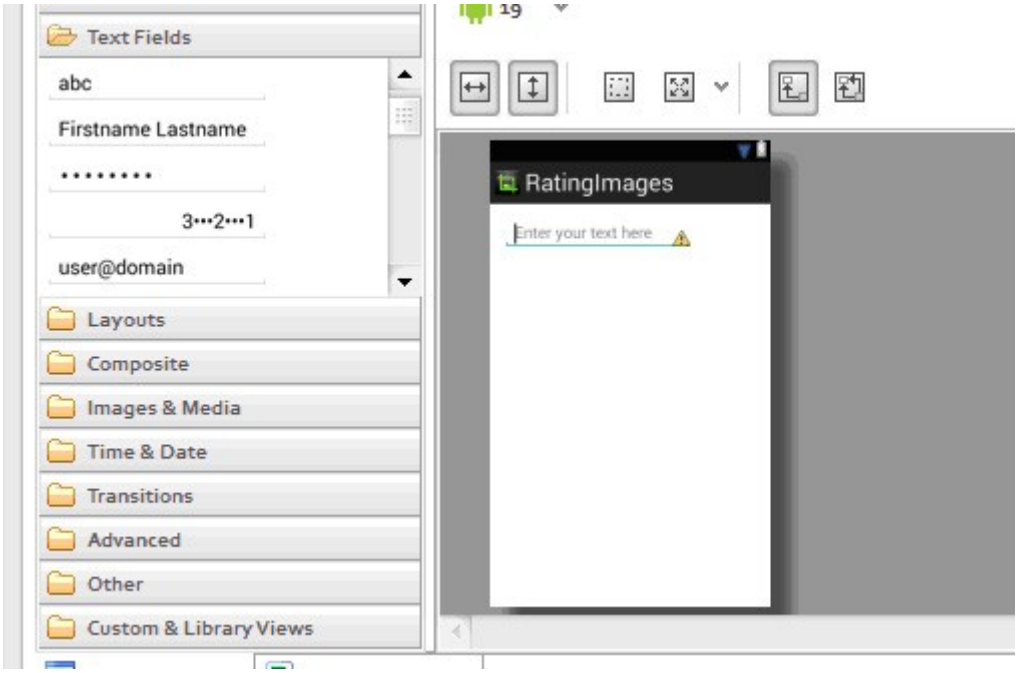

Рис. 14. Отображение текста "По умолчанию" в поле ввода

Подробнее о ресурсах - здесь.

Аналогично создадим id для <LinearLayout>

```
android:id="@+id/linear1"
```

Сохраните изменения.

Добавление кнопки

Теперь добавьте <Button> в макет после элемента <EditText>:

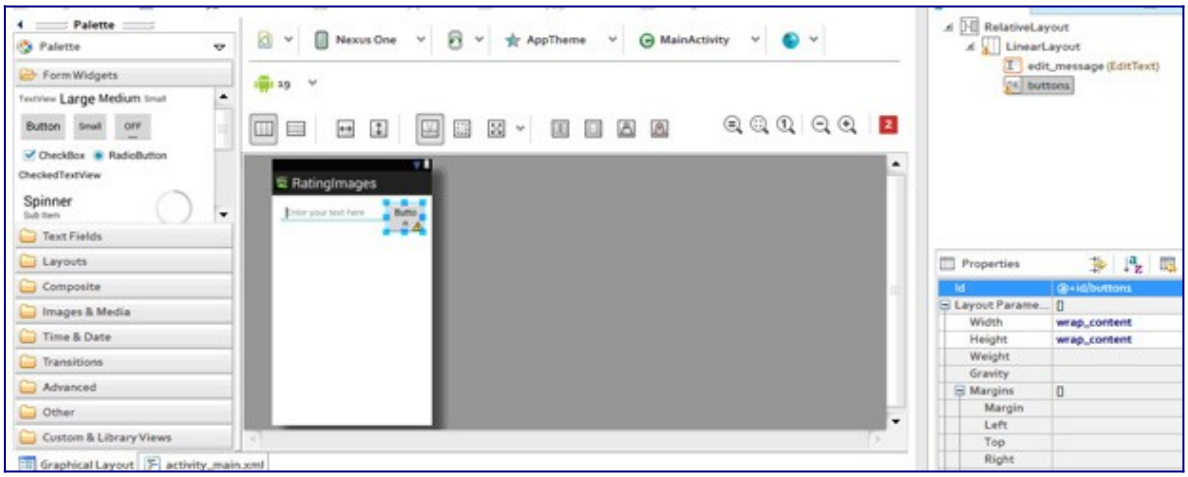

Рис. 15. Новая кнопка

Чтобы кнопка трансформировалась в соответствии с текстом кнопки, ширина и высота должны быть установлены во "wrap_content".

Теперь поменяем надпись на кнопке на "**Go!**" с помощью ссылки на ресурс в

XML-коде главной активности и добавления одного ресурса в файл **strings.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">RatingImages</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="edit_message">Enter your text here</string>
    <string name="button_send">Go!</string>

</resources>
```

Рис. 16. Редактор XML-кода файла strings.xml

Сохраните.

На графическом редакторе главной активности будут изменения:

Рис. 17. Кнопка приняла новую форму, в соответствии с надписью на ней

Теперь, когда мы поместили два главных представления на <LinearLayout> элемент, настало время добавить ещё два параметра для этого элемента.

Речь идет о "приращении" правого и левого краёв лейаута к правому и левому краям <RelativeLayout> элемента соответственно. А сделать это проще простого - просто потяните мышкой один край к другому!

Рис. 18. "Приращение" правого края

Зелёные стрелки по краям дают понять, к какому элементу удалось прицепить край:

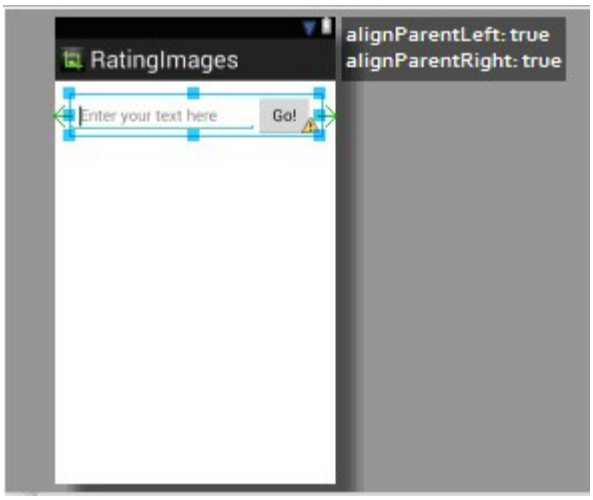

Рис. 19. <LinearLayout> после выравнивания

И, конечно, это отобразится в свойствах:

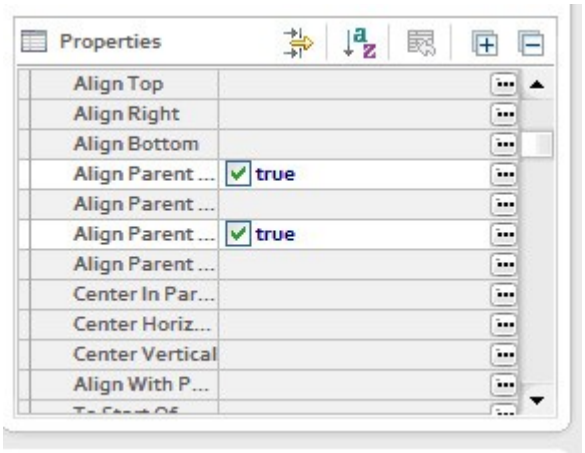

Рис. 20. Свойства <LinearLayout>

Смена фона

Идём дальше - попробуем поменять фон.

Чтобы изменить цвет фона на чёрный, нужно в XML-коде главной активности написать одну строку в блоке <RelativeLayout> элемента:

```
android:background="#000000".
```

Сохраните и проверьте результат, открыв графический редактор.

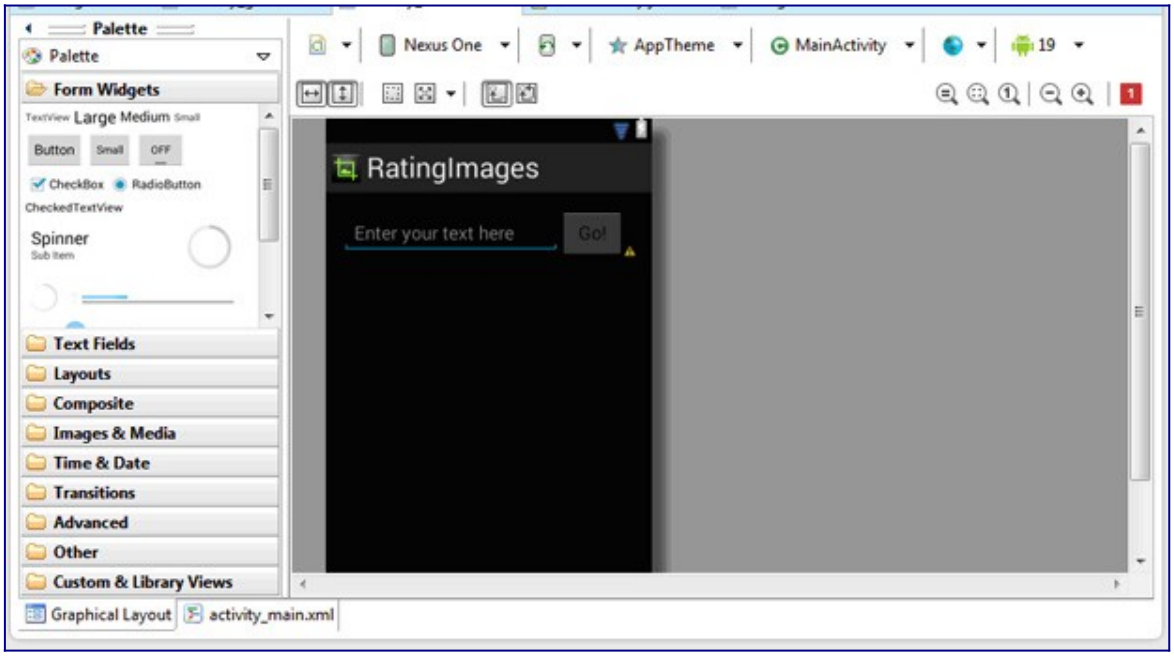

Рис. 21. Фон стал чёрным

Красиво, но скучно. Как сделать фон ещё интереснее? Поместить на него рисунок!

Для этого сначала в папке **res/** создадим папку **drawable/**

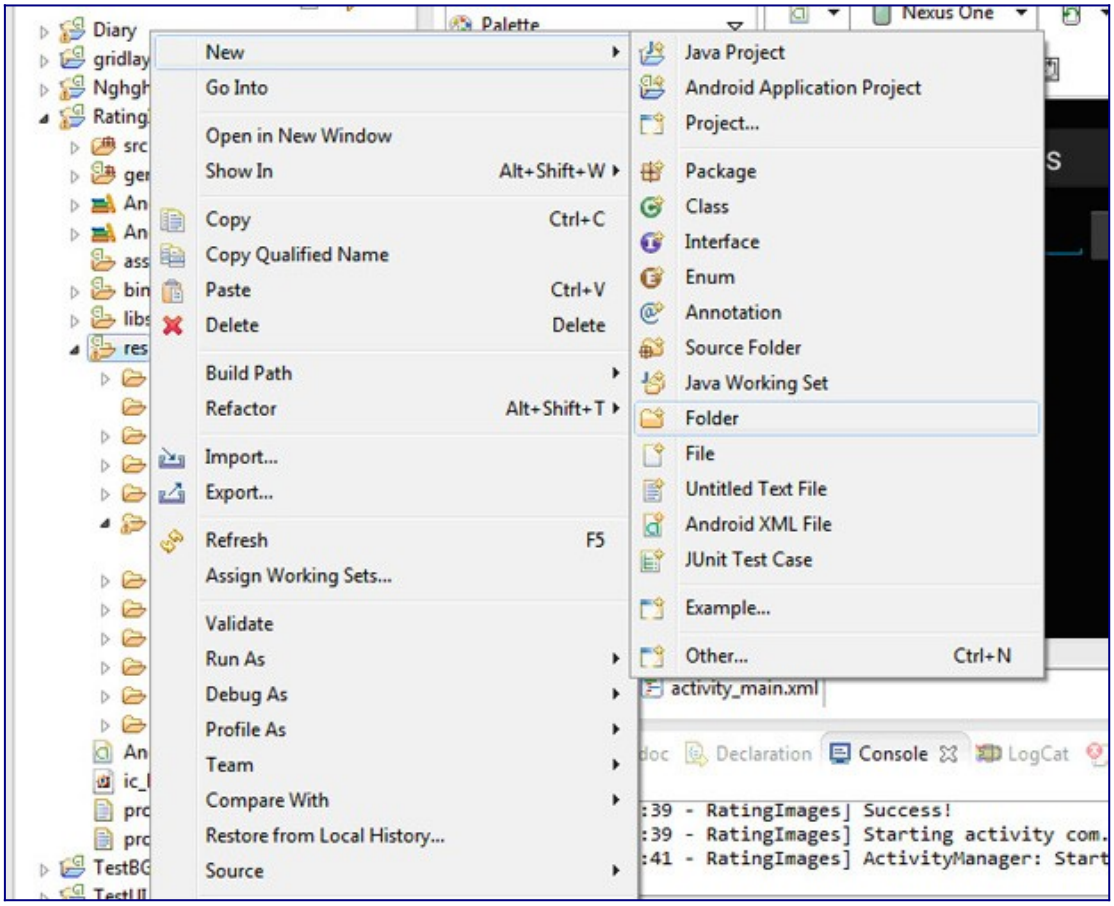

Рис. 22. Создание папки

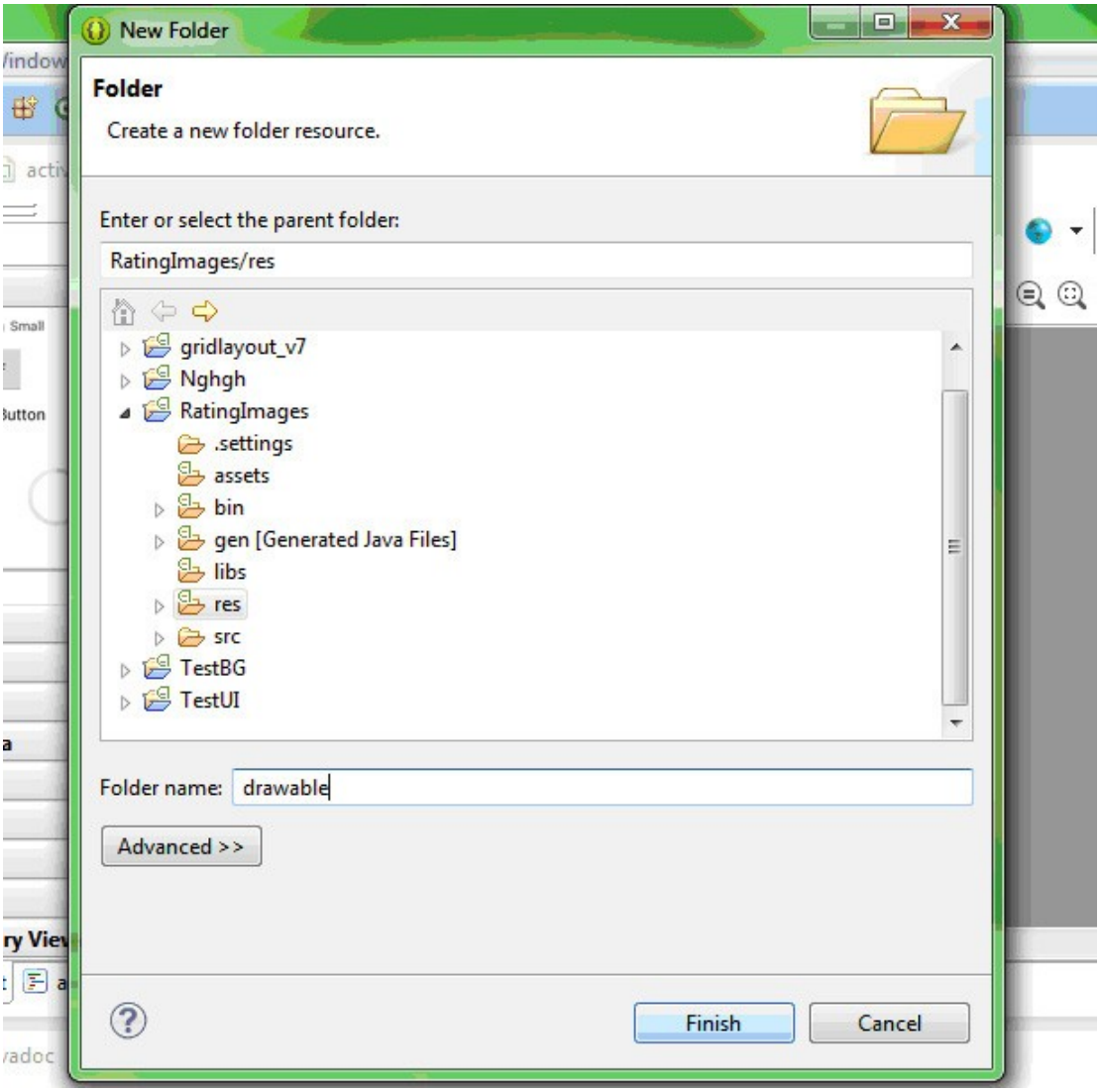

Рис. 23. Создание папки: имя папки

После того, как папка создана, нужно положить в эту папку изображение - картинка называется **got.png**:

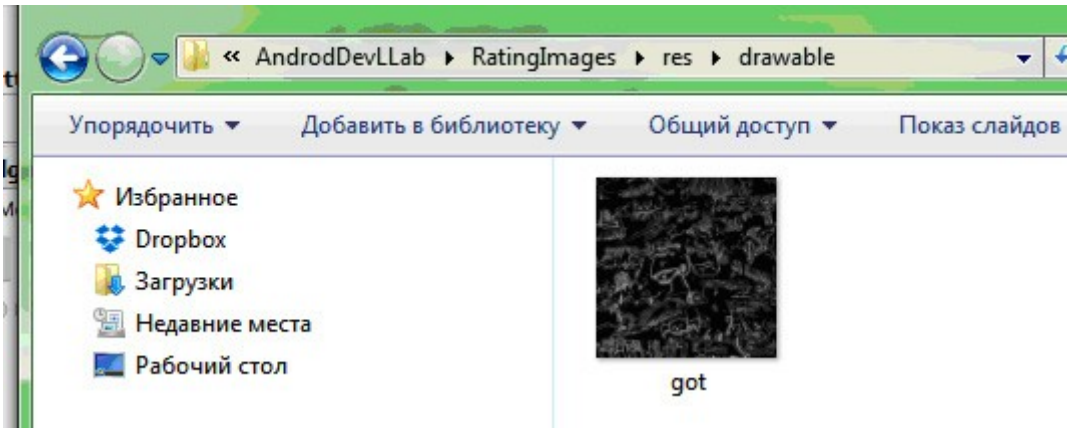

Рис. 24. Изображение в папке drawable/

После этого в папке **drawable/** нужно создать файл **background.xml**, важно при создании выбрать параметр bitmap.

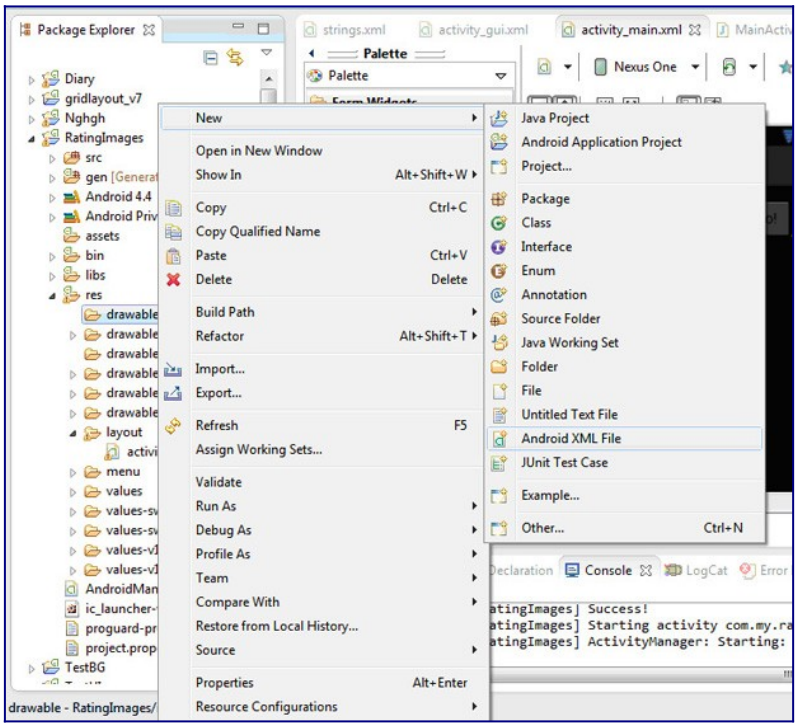

Рис. 25. Создание нового XML-файла

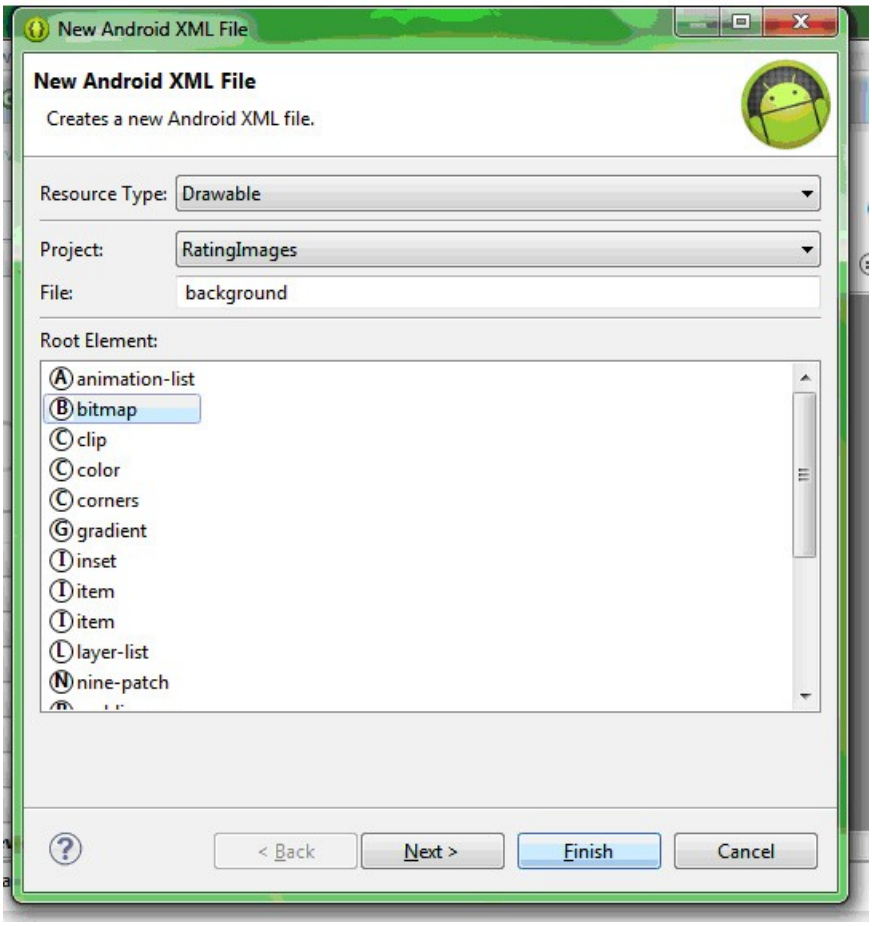

Рис. 26. Создание нового XML-файла

Как только новый файл открылся, пропишем в него одну строчку, с указанием на то, откуда и какой файл использовать:

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
        android:src="@drawable/got">
</bitmap>
```

Вернемся в редактор XML-кода, туда, где прописывали цвет фона.

Вместо строки android:background="#000000" напишем ссылку на XML-файл android:background="@drawable/background".

Сохраняем и видим результат:

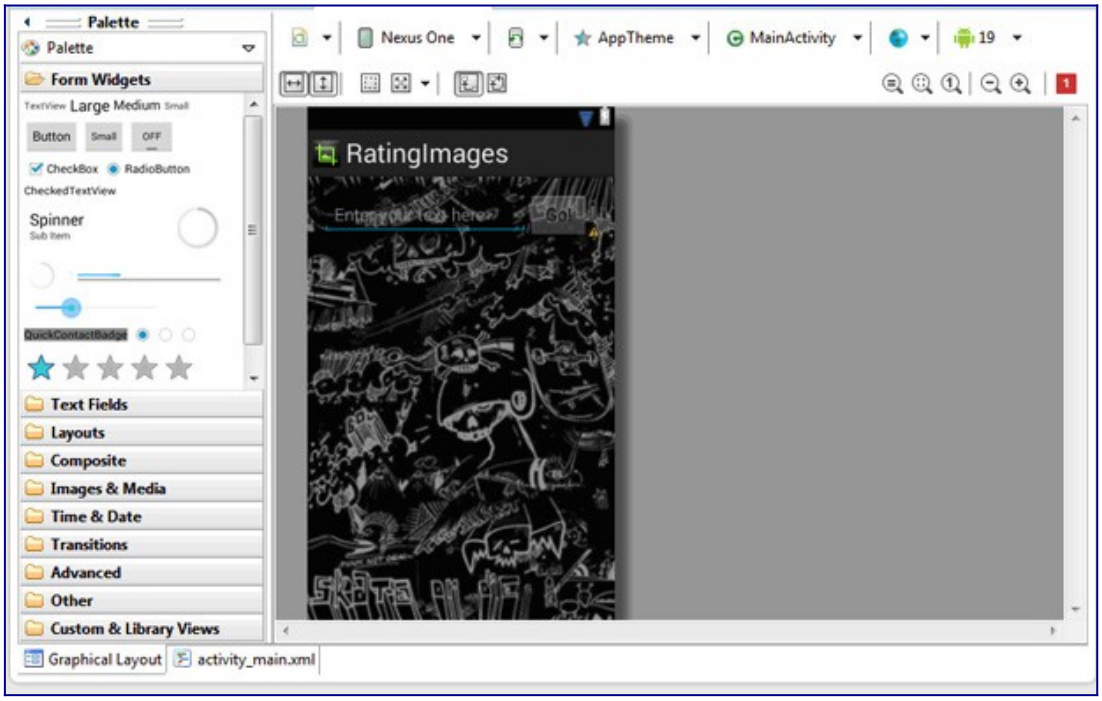

Рис. 27. Новый фон

Несомненно, фон смотрится хорошо, но очевидно, что кнопка и поле ввода просто затерялись, а это значит, что для этого приложения такой фон не подходит. Можно продолжить подбирать изображения на фон, но лучше создать черепичную заливку небольшим изображением. На http://subtlepatterns.com/ сайте можно найти узор на любой вкус!

Когда вы выбрали узор и скачали его, скопируйте изображение в папку **drawable/**.

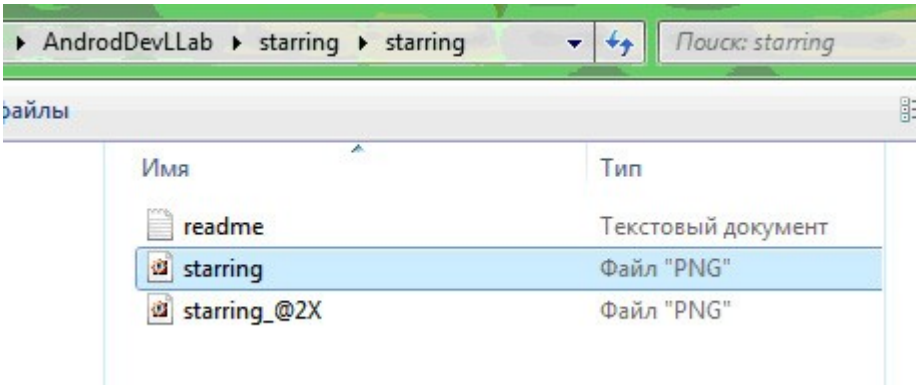

Рис. 28. Копирование изображения

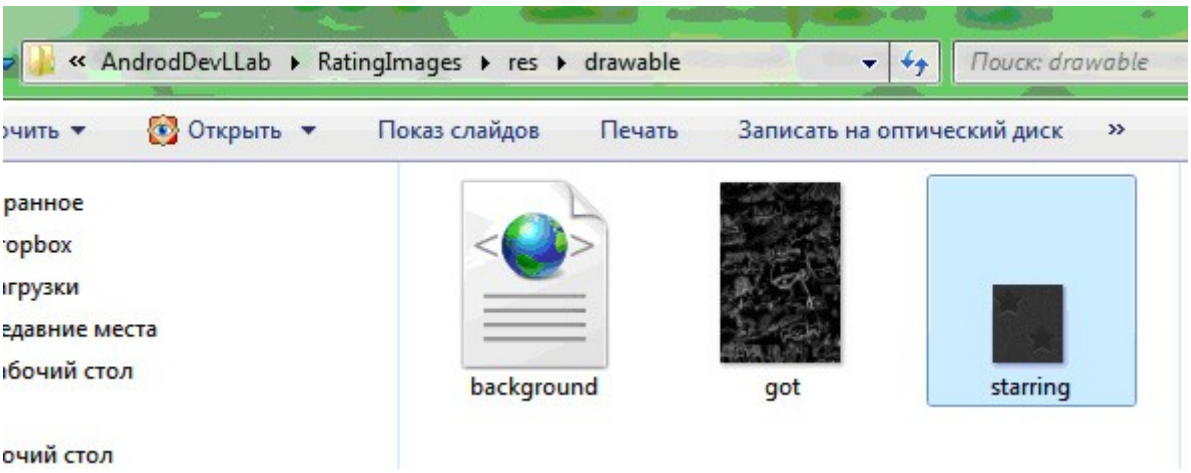

Рис. 29. Скопированное изображение

Теперь немного изменим файл **background.xml**.

Во-первых, нужно изменить имя изображение со старого на новое.

```
android:src="@drawable/starring"
```

Во-вторых, добавим такую строчку:

```
android:tileMode="repeat"
```

Сохраняем.

Атрибут android:tileMode задает тип заполнения, в данном случае простое повторение исходного изображения. Кроме repeat возможны варианты clamp и mirror. Помните, что данный приём применим только к bitmap, к фигурам, созданным при помощи XML, применить данную операцию нельзя.

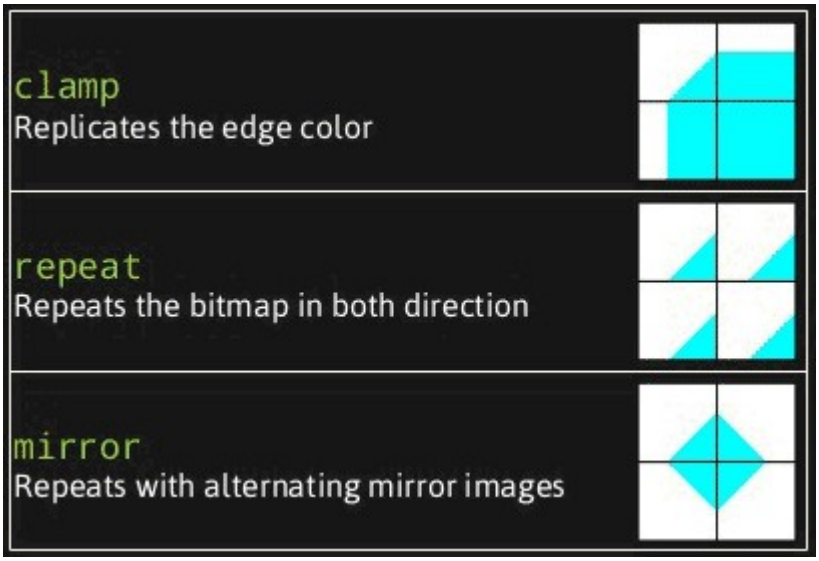

Рис. 30. Варианты заполнения

Настало время посмотреть, что из этого получилось:

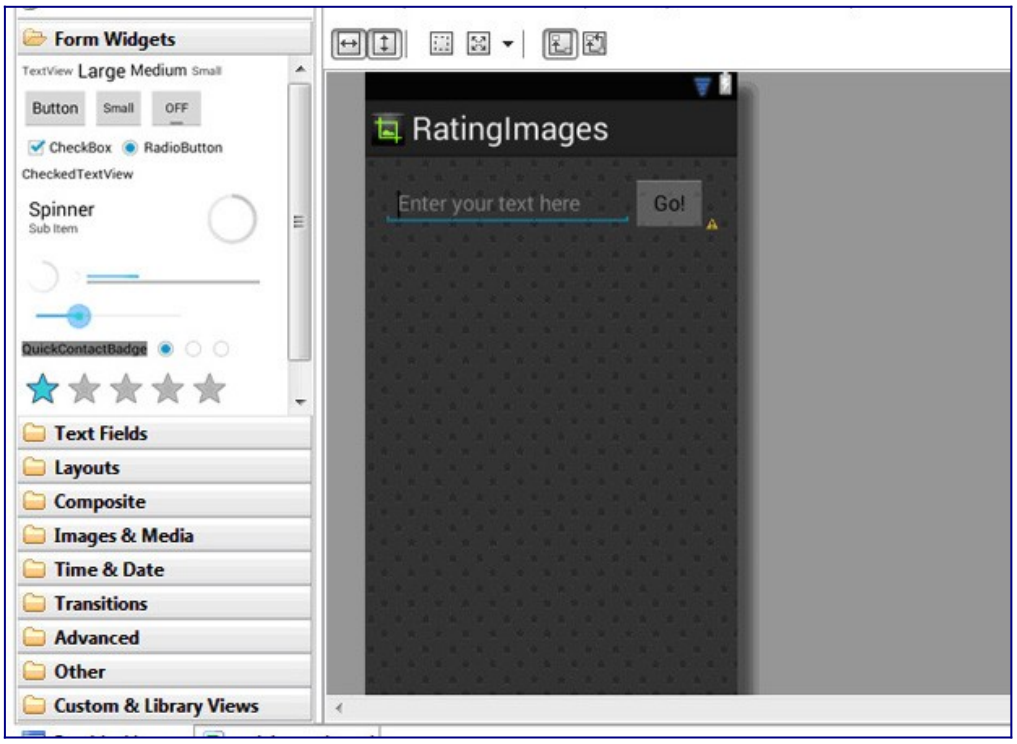

Рис. 31. Фон из звездочек

В <RelativeLayout> стоит добавлять android:background только в том случае, если вы хотите неподвижный фон, а в <ScrollView> чтобы фон прокручивался вместе с контентом.

Если вы выбрали тёмный фон, то стоит поменять цвет текста, вводимого в поле ввода, например на белый.

Для этого в блок <EditText> добавим строчку

```
android:textColor="#ffffff"
```

Область просмотра изображений

Теперь займемся созданием области отображения изображений, которые пользователь будет оценивать.

Добавьте "рамку" <FrameLayout> в <RelativeLayout>:

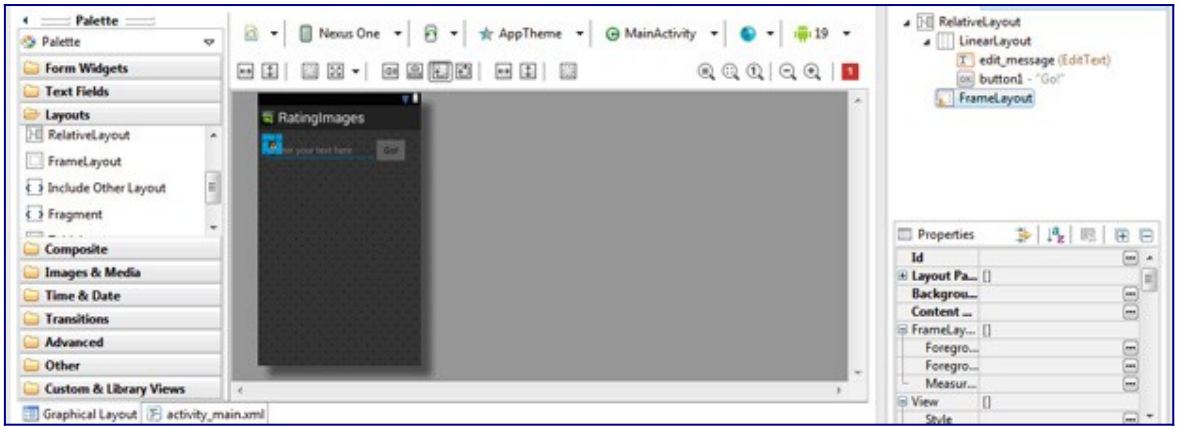

Рис. 32. Добавление "рамки"

Обратите внимание, что предупреждающий жёлтый треугольник исчез, но появился у нового элемента, и это значит, что всё идёт по плану.

Укажем этому элементу ширину, высоту и id. Предупреждение должно пропасть.

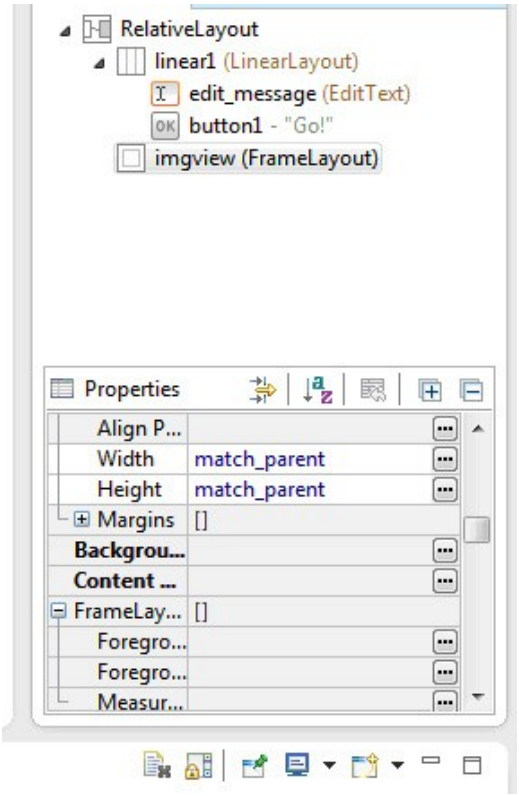

Рис. 33. Свойства "рамки"

Сейчас **FrameLayout** "заезжает" на поле ввода и кнопку. Чтобы исправить это, нужно задать для "рамки" параметр, который будет следить, чтобы "рамка" находилась ниже, чем поле ввода и кнопка.

Выберите из списка **Outline** "рамку" и кликом правой кнопки мыши вызовите контекстное меню: **Other Properties -> Layout Parametrs -> Layout Below…**

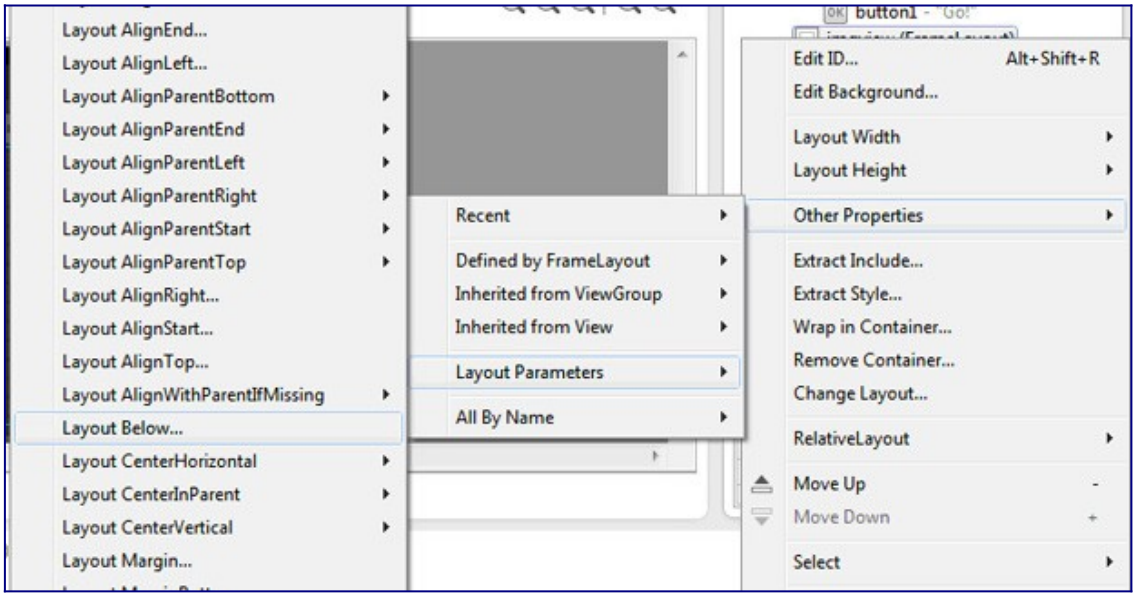

Рис. 34. Контекстное меню "рамки"

Появится окно. В нём выбираем из списка "**ID**" имя лейаута, на котором находятся поле ввода и кнопка:

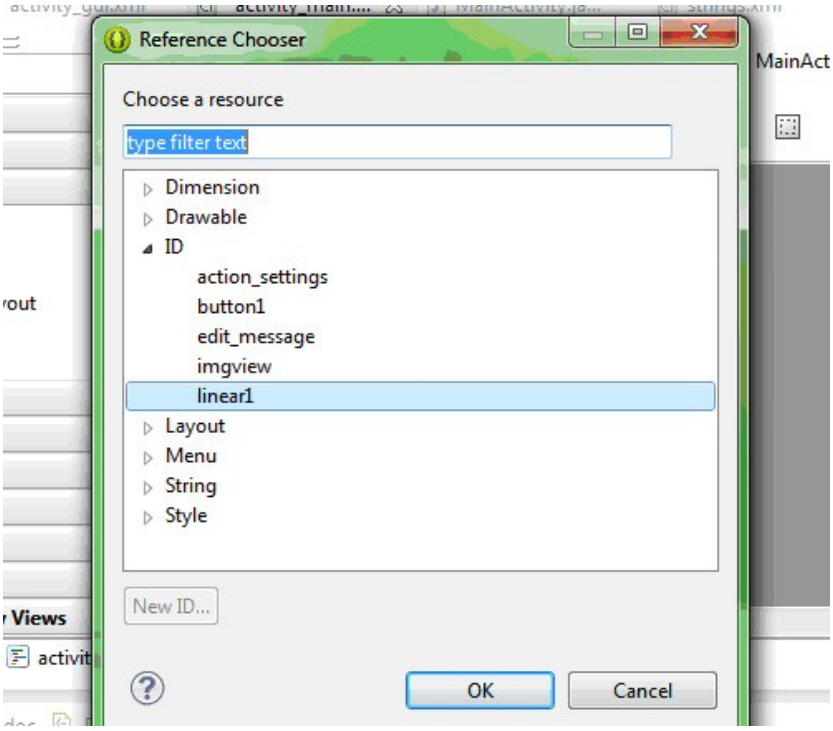

Рис. 35. Выбор элемента, ниже которого должна быть "рамка"

Жмём **OK** и "рамка" примет вид:

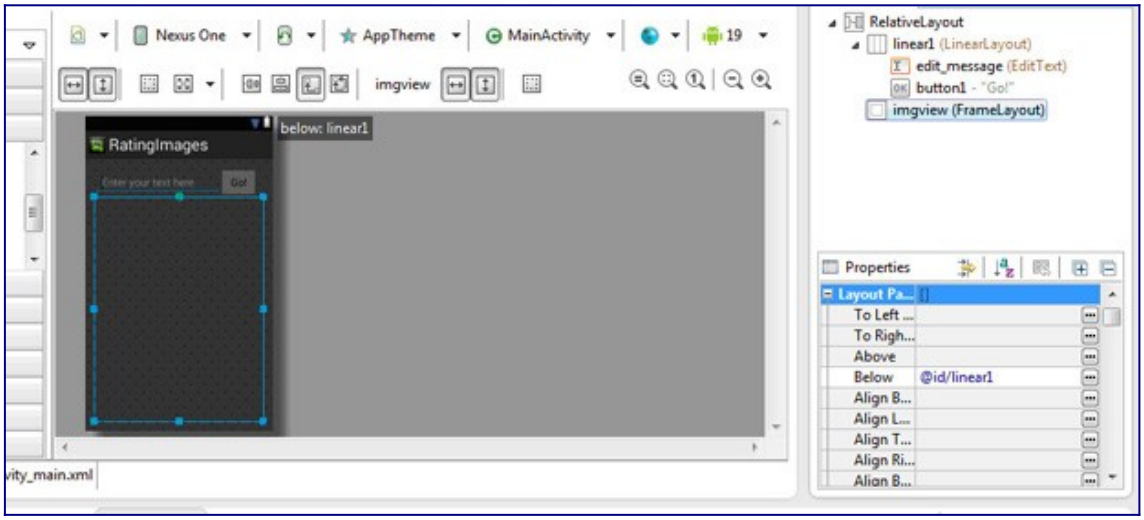

Рис. 36. Новое расположение "рамки"

Теперь для наглядности поместим туда изображение.

Сначала поместим само изображение в папку **res/drawable/** и обновим её.

Это нужно для того, чтобы новые данные загрузились в проект.

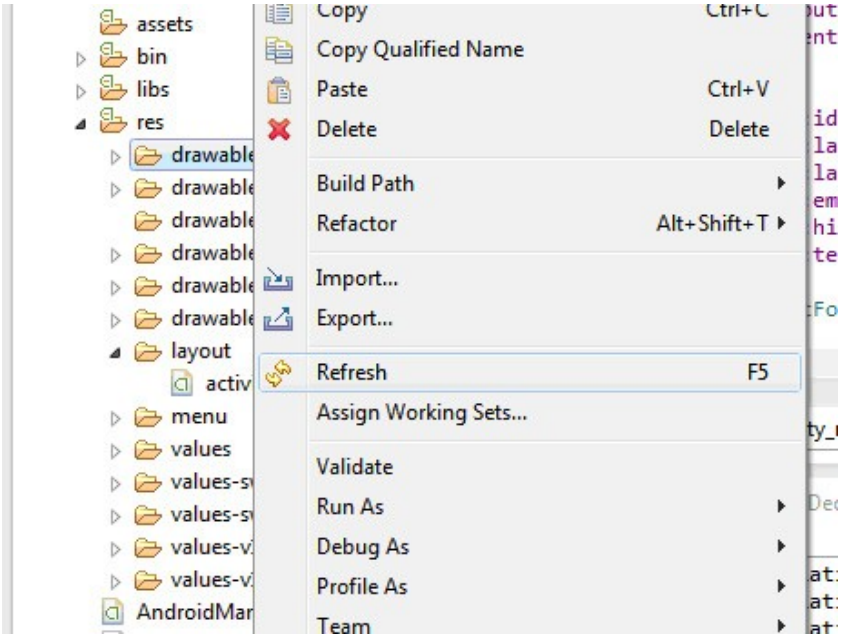

Рис. 37. Обновление папки

Теперь поместим внутрь "рамки" <ImageView>

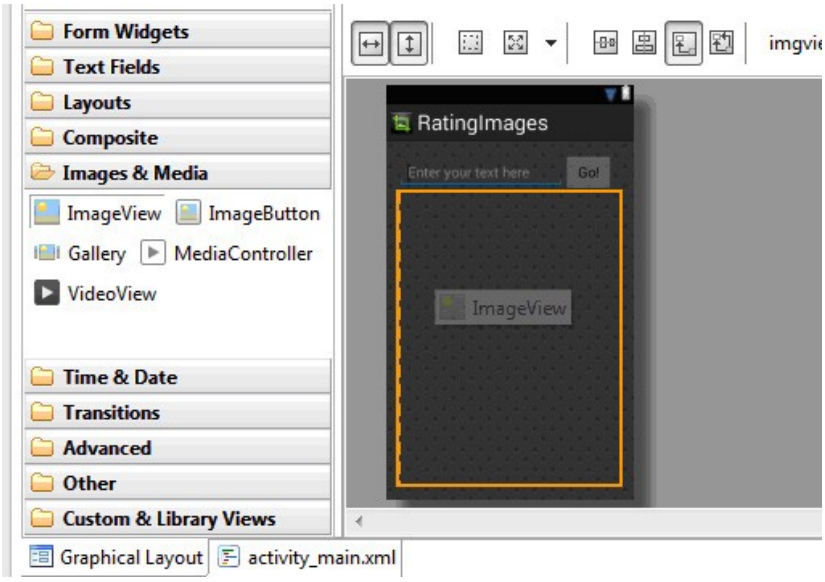

Рис. 38. Перемещение <ImageView>

В появившемся окне выбираем нужное нам изображение и жмём **OK**

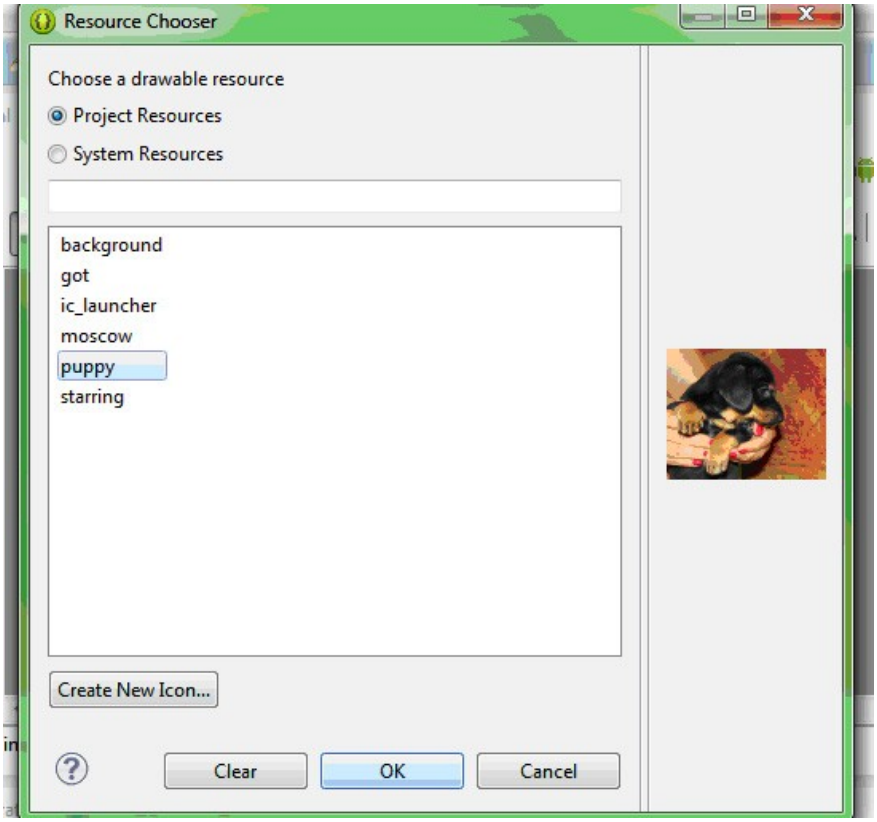

Рис. 39. Выбор файла

Изображение должно появится на активности:

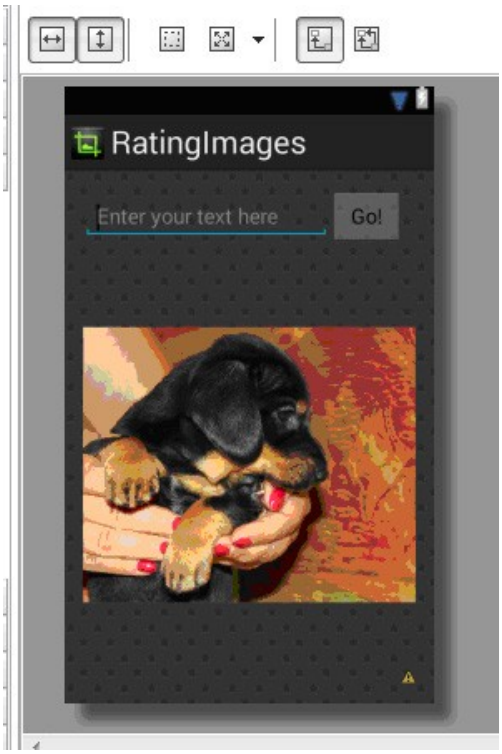

Рис. 40. Визуализация изображения

Вместе с изображением появился новый предупреждающий знак.

В описании предупреждения сказано: "**[Accessibility] Missing contentDescription attribute on image**". Давайте разберёмся, что это означает.

Атрибут android:contentDescription используется в программах, которые поддерживают специальные возможности для людей с нарушениями зрения и слуха. То есть, текст, прописанный в этом атрибуте, не будет показан на экране, но при включенной в "Специальных возможностях" услуге "звуковые подсказки" этот текст будет проговариваться, когда пользователь переходит к этому элементу управления.

Этот атрибут можно задать для ImageButton, ImageView и CheckBox, и задавать его или нет - дело каждого.

Свойства у изображения должны быть следующими:

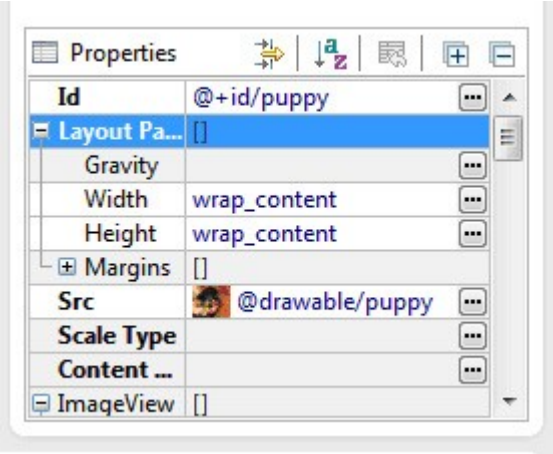

Рис. 41. Свойства изображения

Кнопки "like" и "dislike"

Создадим кнопки оценивания.

Для этого добавьте на форму <RelativeLayout>, задайте для него ширину и высоту wrap_content, и укажите id.

Снова в папку **res/drawable/** нужно добавить файлы. Найдите изображения "Палец вверх" и "Палец вниз", и поместите их в эту папку, после чего обновите её.

Изображения и другие полезные файлы можно скачать по адресу http://developer.android.com/design/downloads/index.html.

Добавьте <ImageButton>, выберите изображение "Палец вверх", и переместите <RelativeLayout> в такое положение:

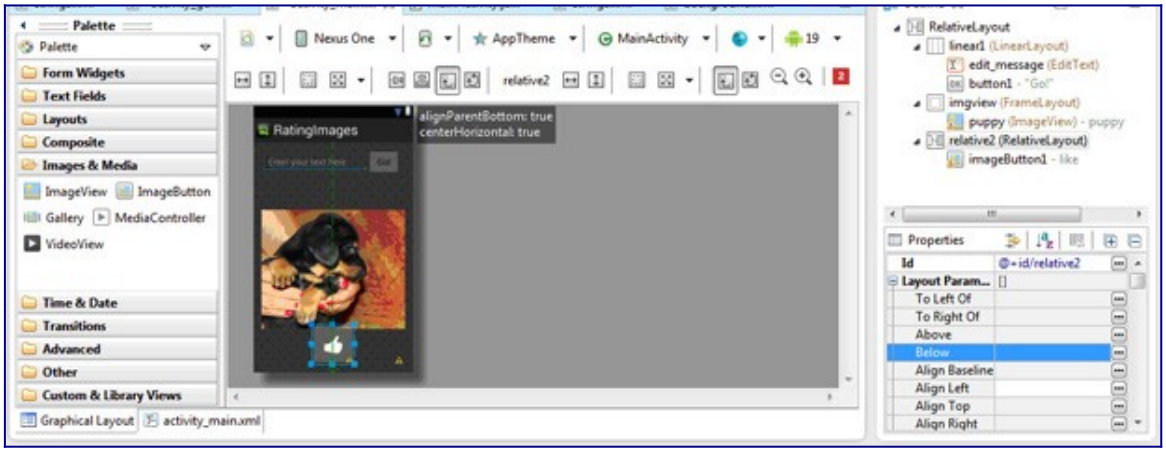

Рис. 42. Новая кнопка

Укажите "рамке" быть выше, чем поле с кнопкой оценки:

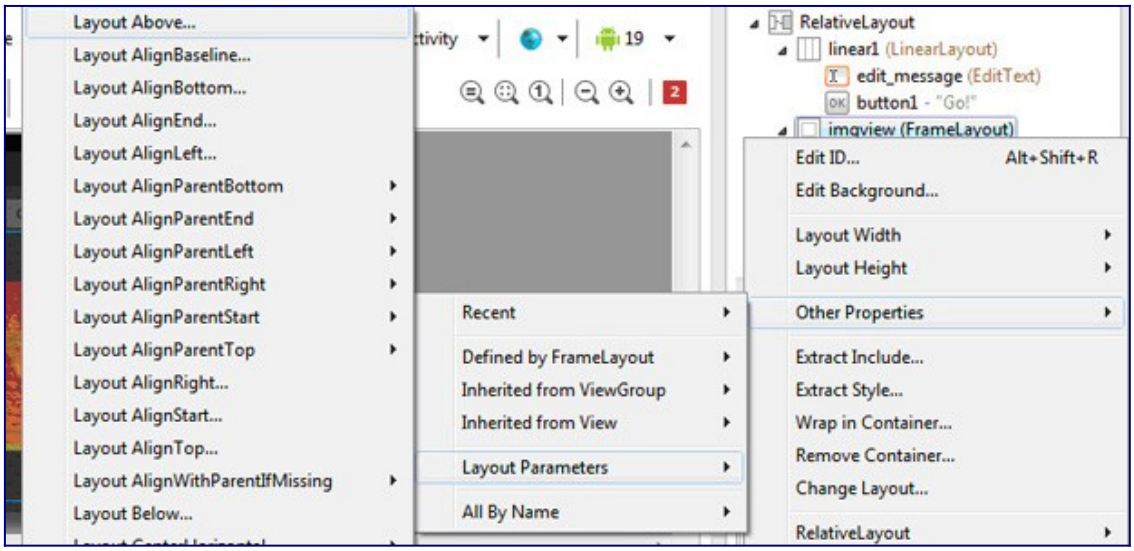

Рис. 43. Контекстное меню "рамки"

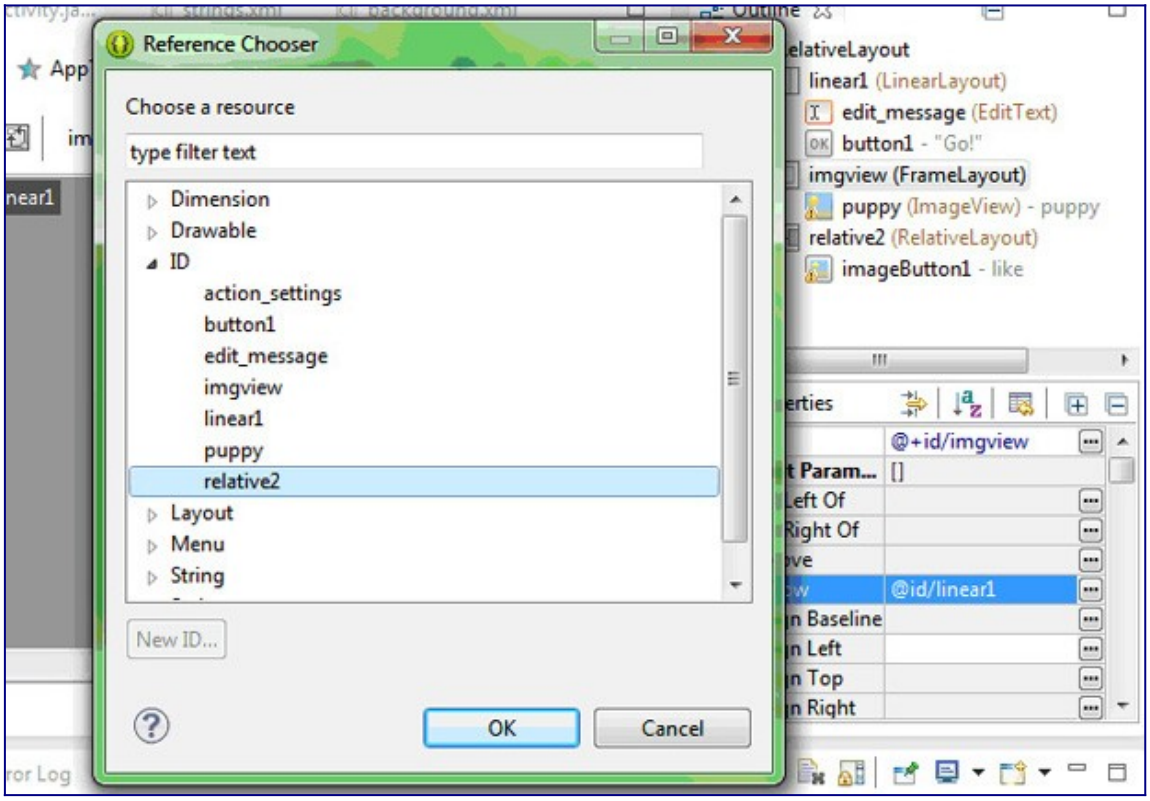

Рис. 44. Выбор элемента, выше которого должна быть "рамка"

В результате должно получиться следующее:

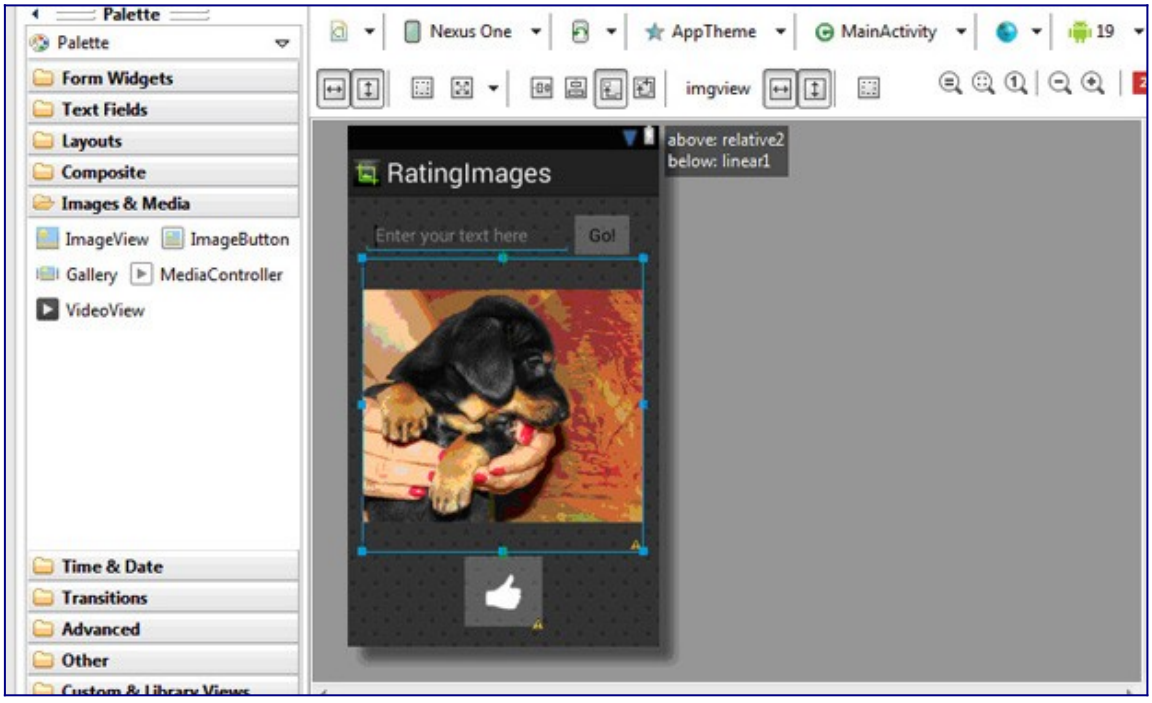

Рис. 45. Новое расположение элементов

Добавьте еще одну кнопку - кнопку "**Палец вниз**". Она "наложилась" на первую кнопку. Чтобы это исправить, проделайте с <RelativeLayout> то же самое, что и с <LinearLayout>: растяните элемент влево и вправо, до получения такого результата:

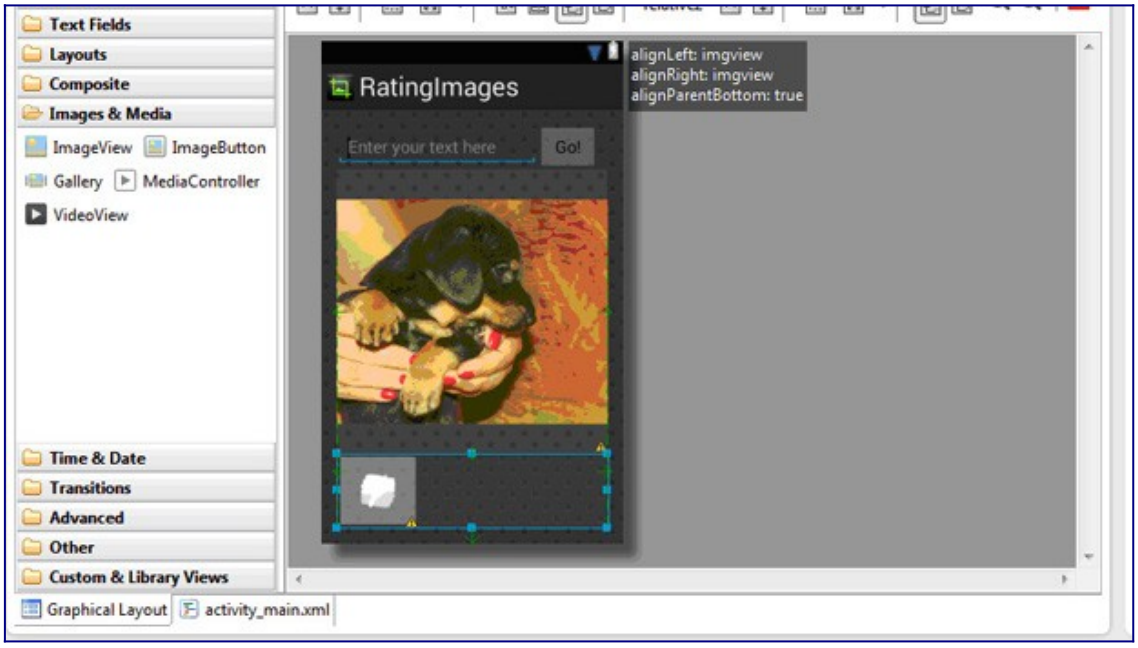

Рис. 46. Выравнивание области с кнопками

Теперь расставьте кнопки по краям так, чтобы они "прикрепились" к краям.

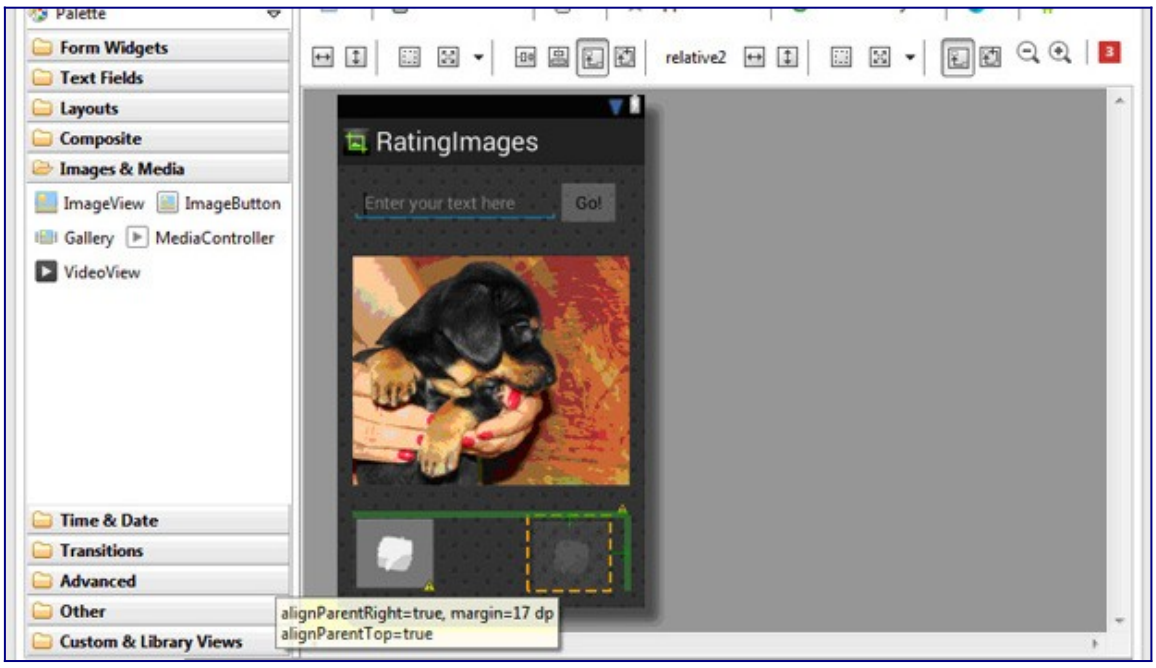

Рис. 47. Выравнивание кнопок

Готово! Теперь можно запустить эмулятор и посмотреть, что получилось.

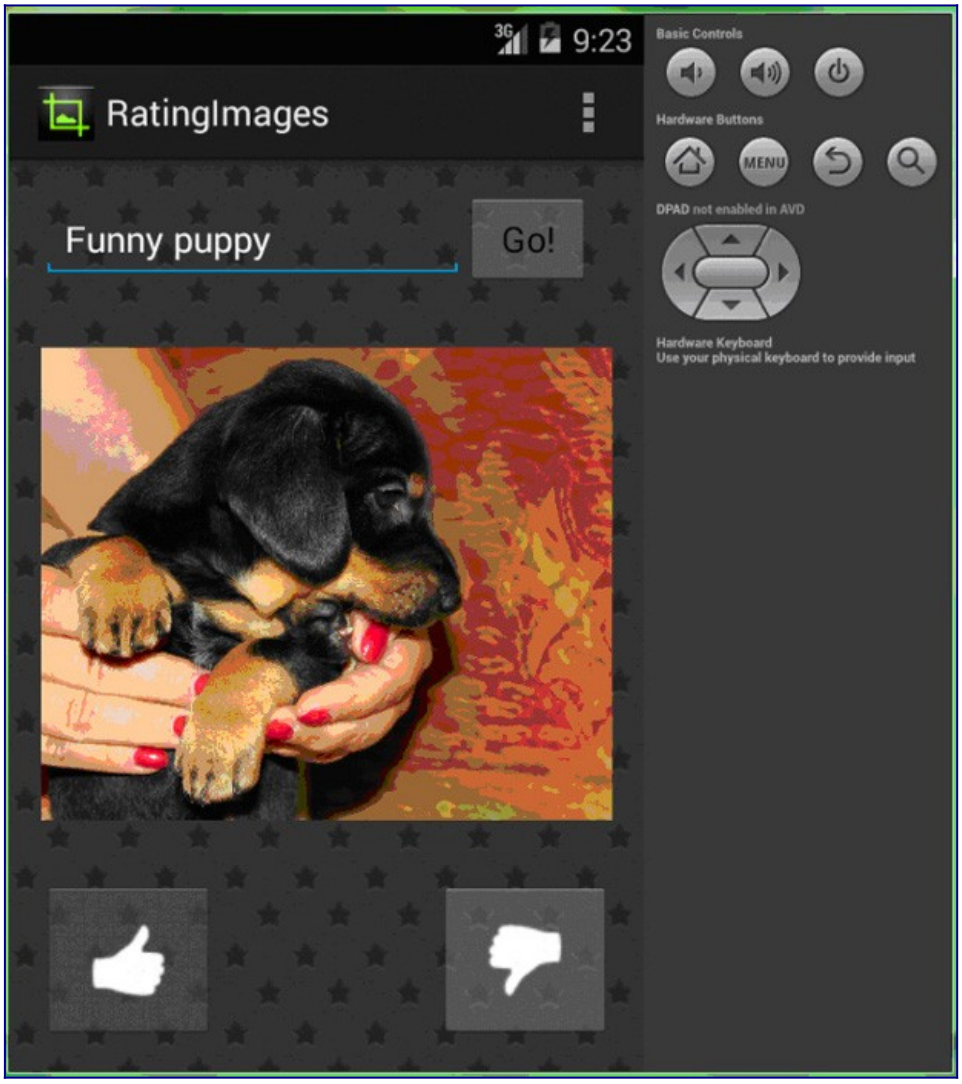

Рис. 3.48. Главная активность

Листинги

1)

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        tools:context=".MainActivity"
        android:background="@drawable/background">

        <LinearLayout
                android:id="@+id/linear1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_alignParentRight="true"
                android:orientation="horizontal" >
```

```
                <EditText
                        android:id="@+id/edit_message"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:ems="10"
                        android:hint="@string/edit_message"
                        android:textColor="#ffffff" >
                        <requestFocus />
                </EditText>

                <Button
                        android:id="@+id/button1"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="@string/button_send"
                />
        </LinearLayout>

        <FrameLayout
                android:id="@+id/imgview"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_above="@+id/relative2"
                android:layout_below="@id/linear1" >
                <ImageView
                        android:id="@+id/puppy"
                        android:layout_width="wrap_content"
                        android:layout_height="match_parent"
                        android:src="@drawable/puppy"
                />
        </FrameLayout>

        <RelativeLayout
                android:id="@+id/relative2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignLeft="@+id/imgview"
                android:layout_alignParentBottom="true"
                android:layout_alignRight="@+id/imgview" >
                <ImageButton
                        android:id="@+id/imageButton2"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_alignParentRight="true"
                        android:layout_alignParentTop="true"
                        android:src="@drawable/dislike"
                />
                <ImageButton
                        android:id="@+id/imageButton1"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_alignParentLeft="true"
                        android:layout_alignParentTop="true"
                        android:src="@drawable/like"
                />
        </RelativeLayout>

</RelativeLayout>
```

2)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

                <string name="app_name">RatingImages</string>
                <string name="action_settings">Settings</string>
                <string name="hello_world">Hello world!</string>
                <string name="edit_message">Enter your text here</string>
                <string name="button_send">Go!</string>

</resources>
```

3)

```
<?xml version="1.0" encoding="utf-8"?>

<bitmap
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:src="@drawable/starring"
        android:tileMode="repeat" >

</bitmap>
```

BUILDINGBLOCKS ИЛИ ЭЛЕМЕНТЫ ДЛЯ ПОСТРОЕНИЯ ИНТЕРФЕЙСА

Среда разработки "**Android Studio**" открывает широкий выбор готовых к использованию элементов для создания выдающихся приложений.

Кнопка (Button) состоит из текста и/или изображения, которые дают понять пользователю, что произойдёт, если нажать на эту кнопку. Важно помнить, что человек по своей природе привык взаимодействовать с объектами, поэтому фон у кнопки абсолютно не обязателен.

Рис. 55a. Кнопка-изображение

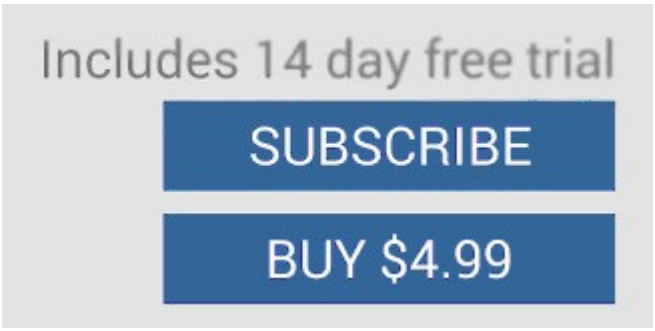

Рис. 55b. Кнопка-текст

Текстовые поля (TextFields)

Текстовые поля позволяют пользователю вводить текст в приложения. Они могут быть однострочными или многострочными. Одно касание текстового поля помещает курсор на поле ввода и автоматически выводит на экран клавиатуру. В дополнение к набору текста на клавиатуре текстовые поля позволяют выделять текст (вырезать, копировать, вставить). Поиск вариантов завершения слова помогает правописанию слов и упрощает поиск контактов в списке.

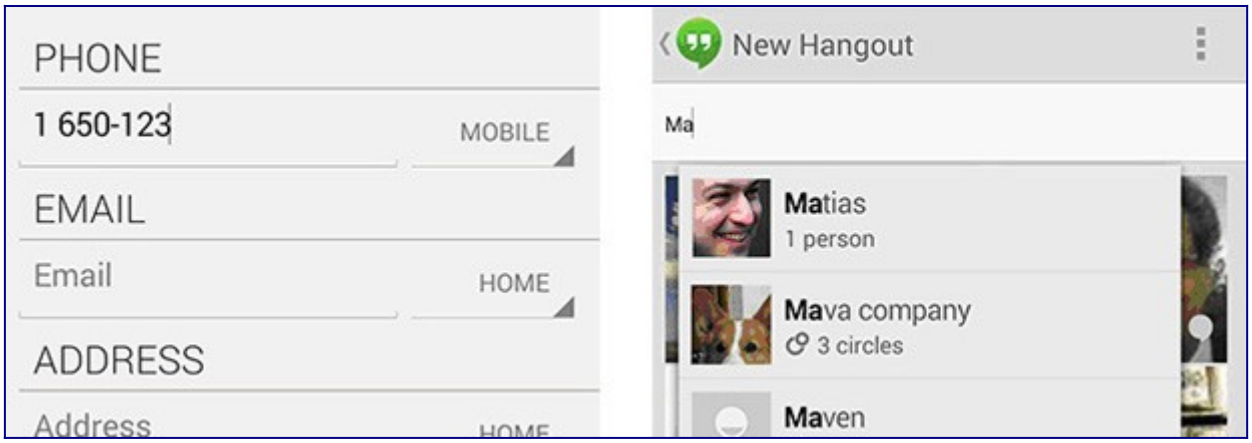

Рис. 56. 50 Текстовые поля, поиск контактов

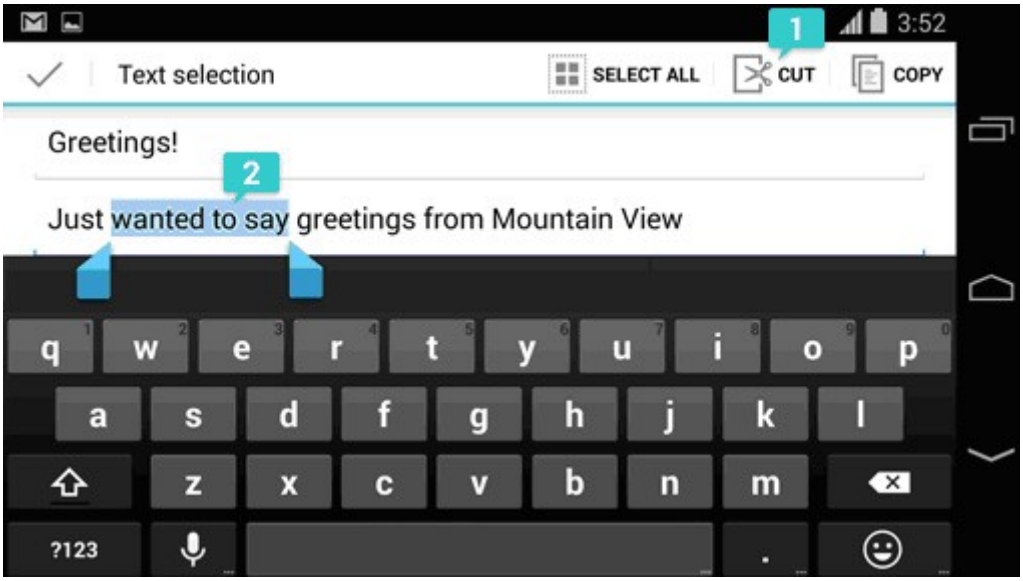

Рис. 57. Выделение текста

Ползунки и слайдеры (Seek Bars and Sliders)

Интерактивные ползунки позволяют выбирать значение из непрерывного или дискретного диапазона значений путем перемещения ползунка. Наименьшее значение находится слева, наибольшее справа.

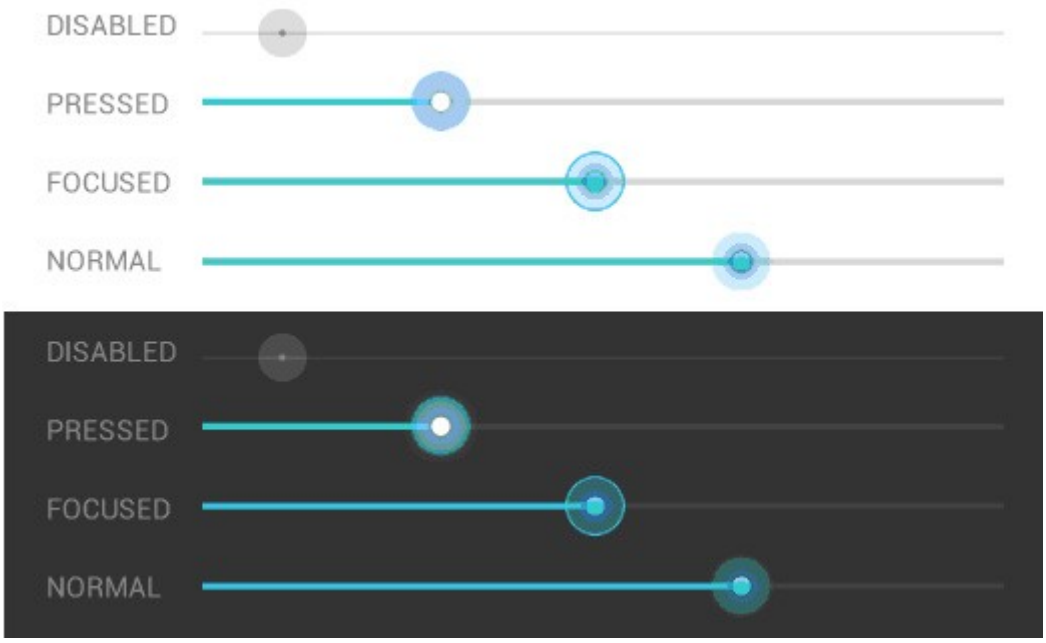

Рис. 58. Ползунки в светлой и темной темах

Интерактивный характер слайдера делает его удобным для настроек, которые отражают уровни интенсивности, такие как громкость, яркость, или насыщенность цвета.

Так, например, уровень громкости можно регулировать кнопками на корпусе устройства или при помощи жеста пальцем по экрану.

Рис. 59. Настройка звука

Прогресс-бары и активности (Progress&Activity)

Прогресс-бары и показатели активности сигнализируют пользователям о происходящем в данный момент времени длительном действии, что означает для пользователя подождать завершения процесса некоторое время.

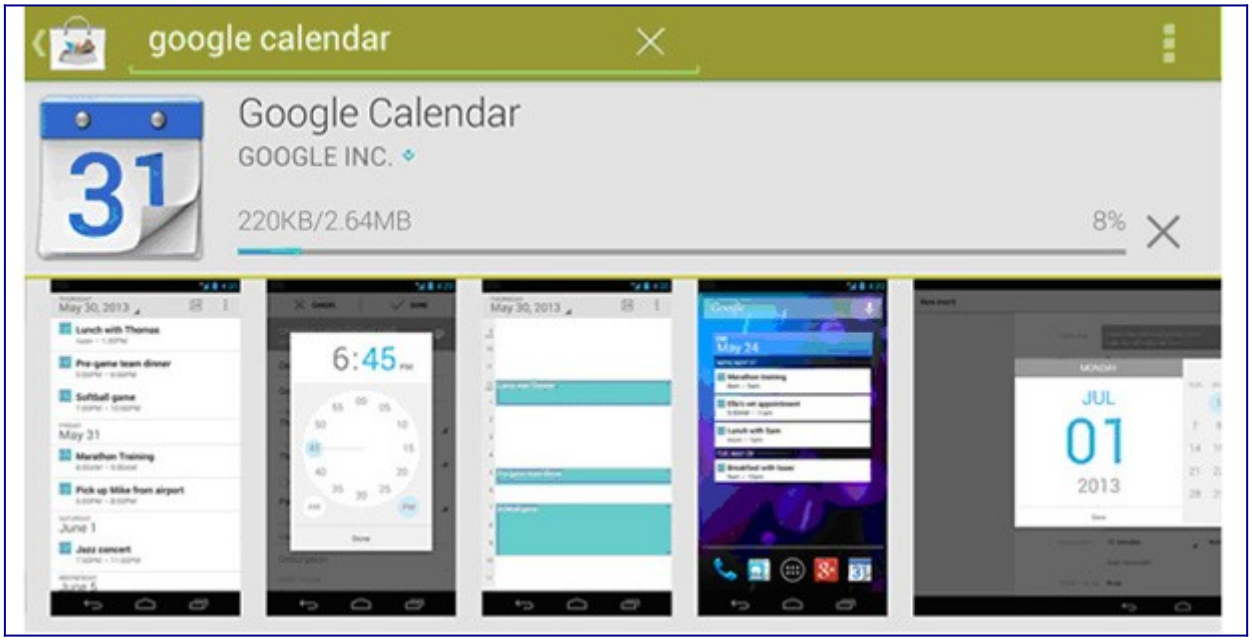

Рис. 60. Процесс загрузки приложения

Переключатели (Switches) позволяют пользователю выбирать параметры.

Есть три вида переключателей: флажки, радио-кнопки, и включение/выключение выключателей.

Флажки используются в том случае, если из предлагаемого списка можно выбрать одновременно несколько вариантов.

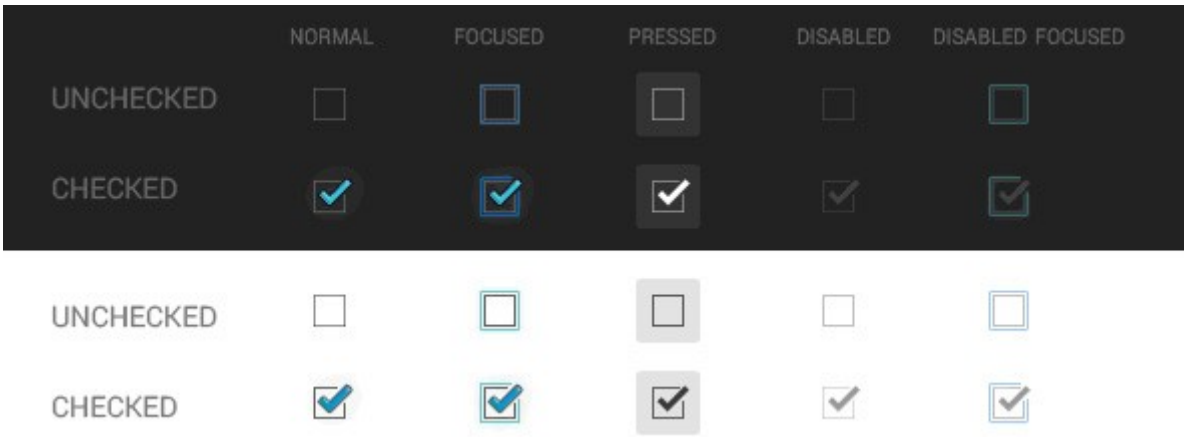

Рис. 61. Флажки

Радио-кнопки позволяют выбрать только один вариант из списка. Радио-кнопки формируются в группы.

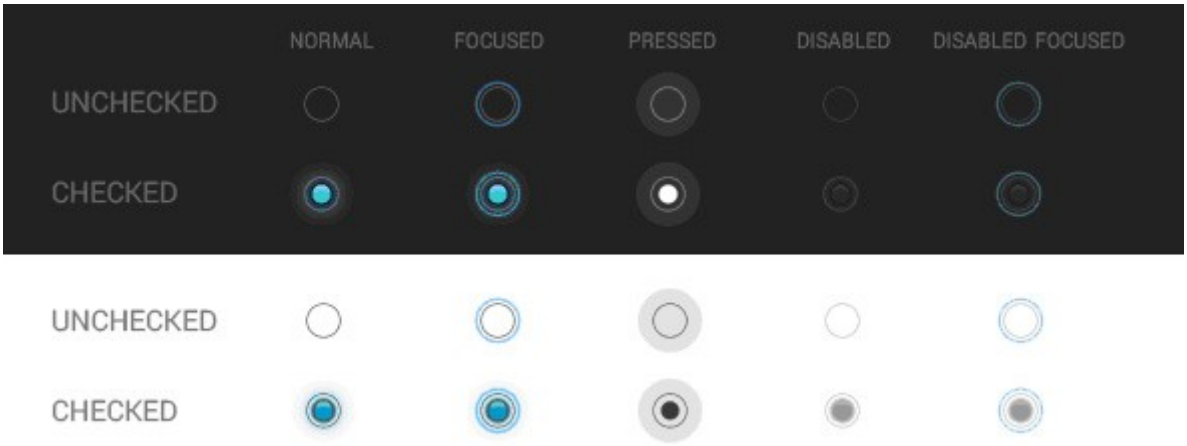

Рис. 62. Радио-кнопки

Выключатели дают возможность сделать флажок более наглядным, применив в качестве основы кнопку-значок, которая может фиксироваться в нажатом состоянии.

Рис. 63. Выключатели

Диалоговые окна (Dialogs) помогают приложению взаимодействовать с пользователем. Это может быть как банальный вопрос с двумя вариантами ответа (OK и Отмена), так и сложные окна с множеством настраиваемых пользователем параметров.

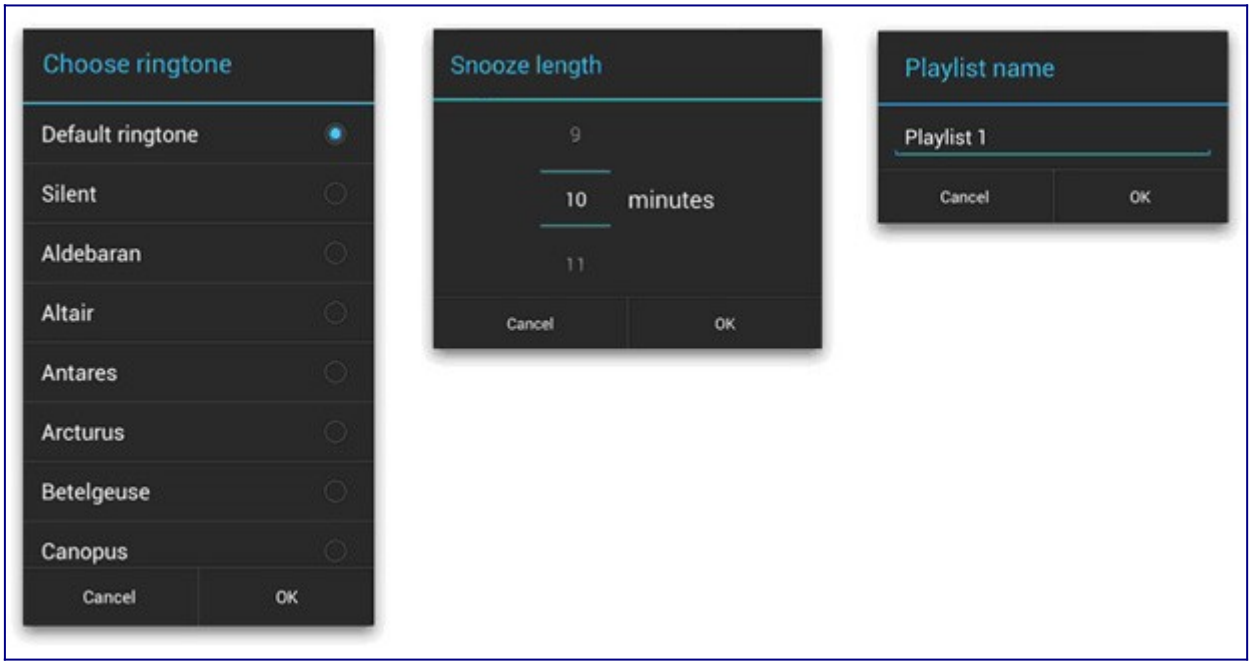

Рис. 64. Примеры диалоговых окон

Средство выбора (Pickers) - это простой способ выбрать одно значение из набора путём последовательного перебора. Удобный перебор предлагаемых значений при помощи клика по стрелочкам или прокрутки, так же можно ввести значение с клавиатуры.

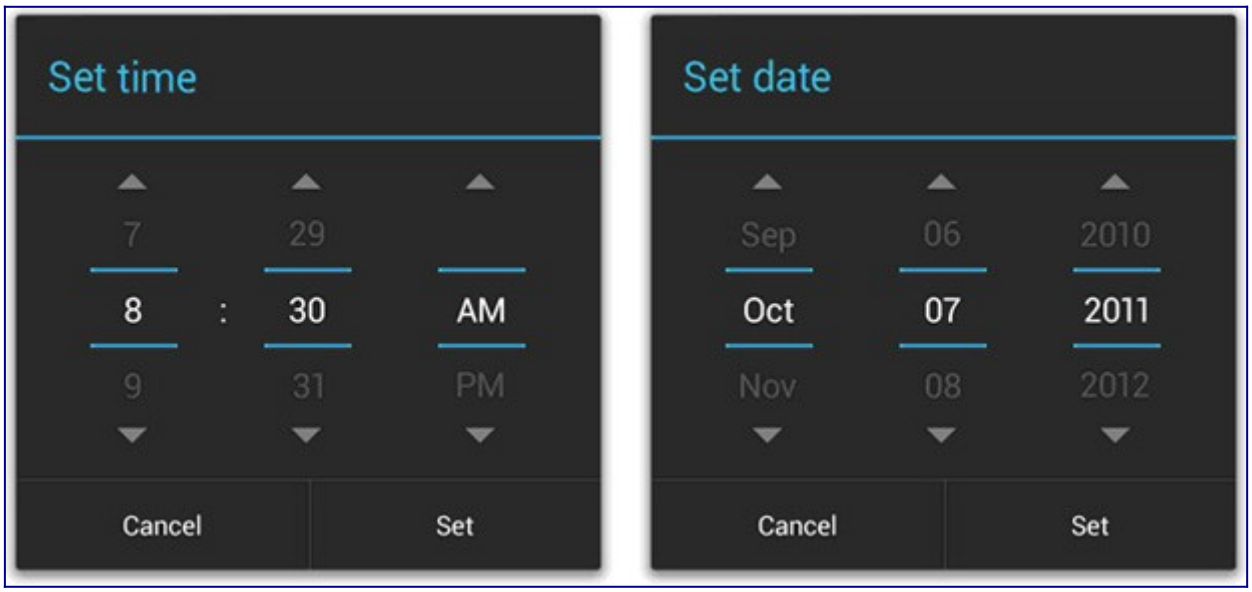

Рис. 65. Примеры пикеров

Прокрутка (Scrolling), как интуитивно ожидаемый элемент, уже давно плотно сидит в головах у разработчиков и пользователей. Плавное или быстрое перемещение по содержимому экрана и контенту, который "вылез" за рамки экрана, осуществляется прокруткой, достаточно лишь провести пальцем по экрану в нужную сторону с желаемой скоростью.

Для того чтобы индикатор прокрутки не занимал место у рабочей области

экрана, после приостановки пользования прокруткой, индикатор исчезает, и появляется снова, стоит только начать "прокручивать" пальцем. Для удобства поиска чего-либо в алфавитном списке, рядом с индикатором прокрутки возникает индекс с указанием на то, в каком разделе находится индикатор.

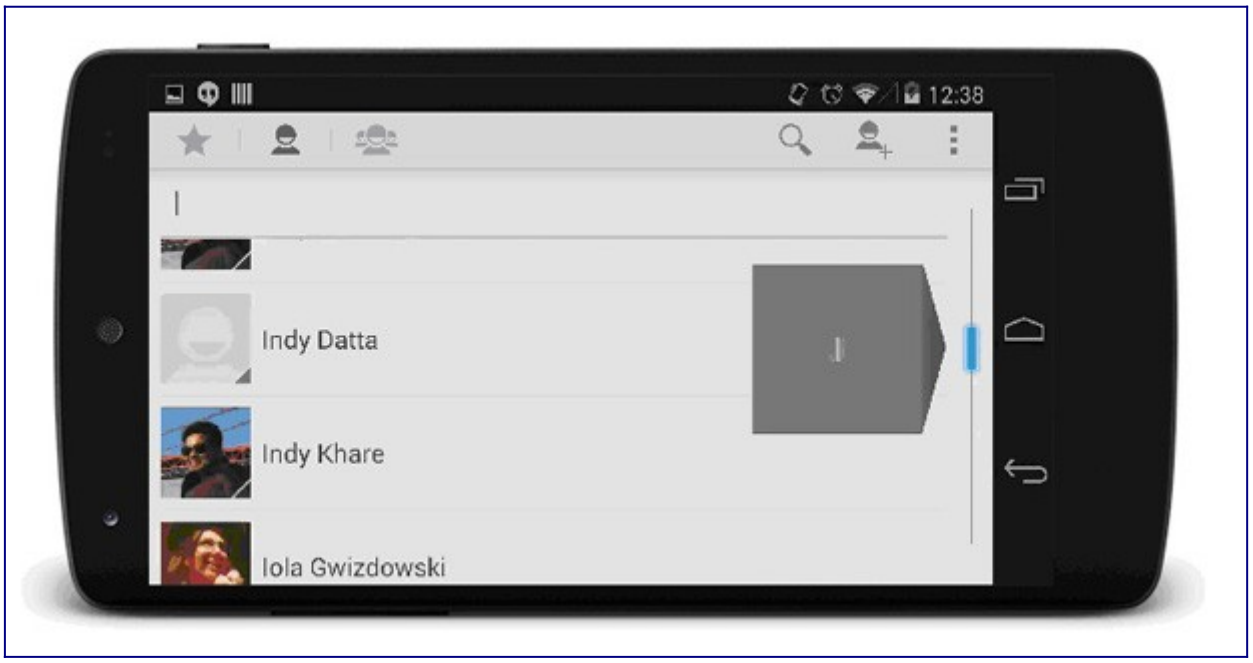

Рис. 66. Индикатор и индекс прокрутки

Выпадающий список (Spinner) обеспечивает удобный способ выбора одного значения из набора.

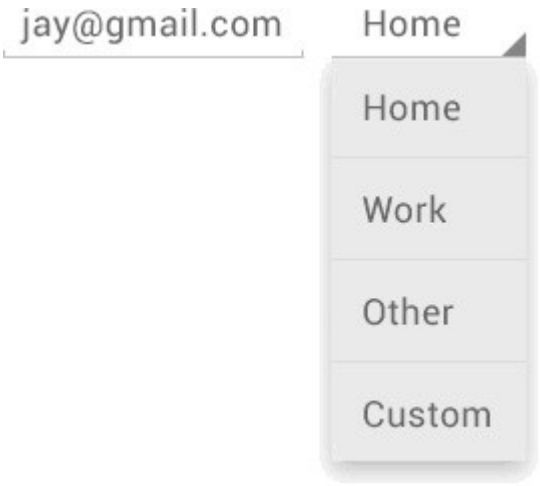

Рис. 67. Спиннер для выбора типа e-mail адреса

Вкладки (Tabs) позволяют легко просматривать и переключаться между различными окнами или функциональными элементами приложения. Согласно материальному дизайну выделены два типа вкладок ScrollableTabs и fixedTabs (см. https://material.io/components/tabs#anatomy). Вкладки, прокручиваемые движением пальца (ScrollableTabs), делают этот процесс еще прощё и приятнее.

Основные вкладки (FixedTabs) удобны при отображении трех и менее вкладок, поскольку ширина вкладок фиксируется по самому длинному названию вкладки.

Рис. 68. Вкладки

Списки (Lists) удобны для представления нескольких позиций информации в вертикальном расположении.

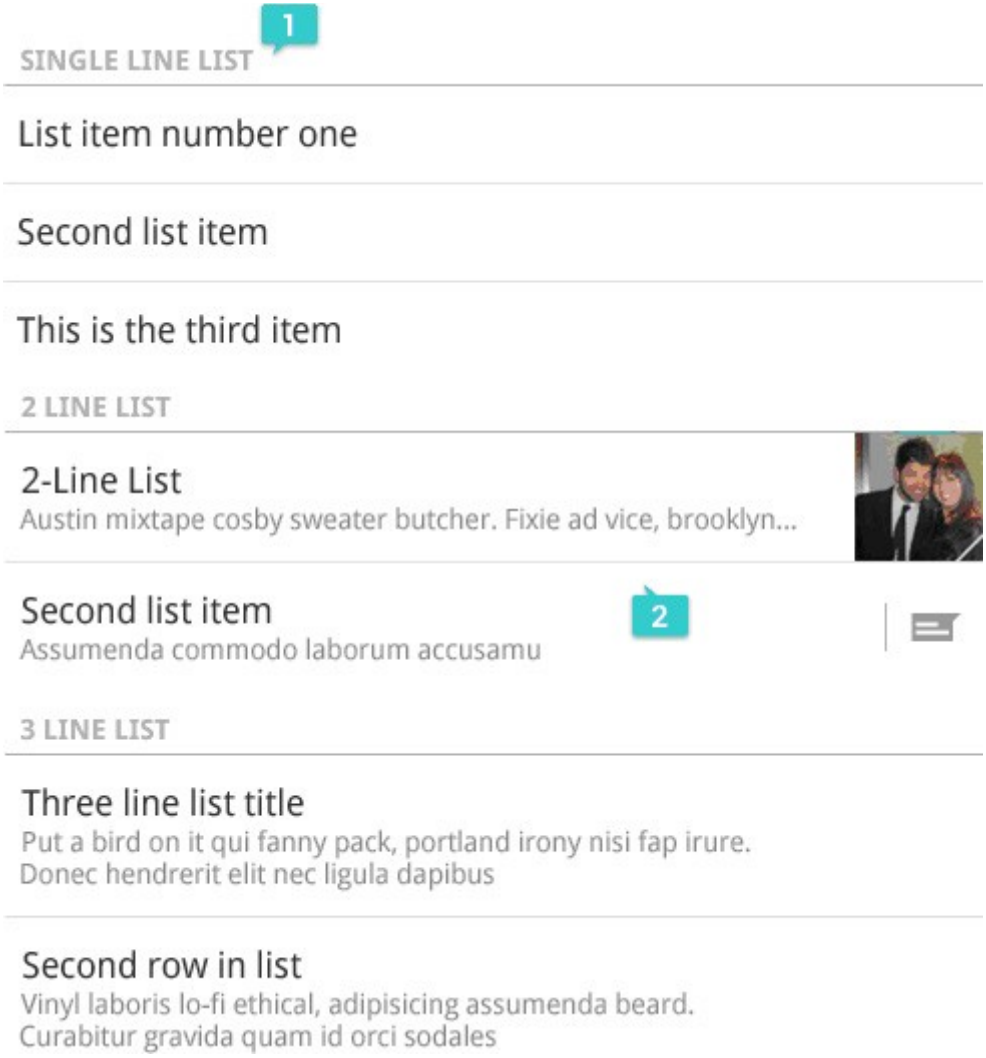

Рис. 69. Список

Сетки (GridLists) - это альтернатива стандартного списка. С помощью сетки можно создавать более удобное представление образных данных, нежели с помощью списка. Ещё одним плюсом сетки является то, что прокрутку можно осуществлять не только в вертикальном, но и в горизонтальном направлении.

Однако нужно стараться избегать создания сетки с возможностью прокрутки в обоих измерениях сразу.

Использование ярлычков для всех панелей сетки и полупрозрачных "занавесок" для неактивных панелей помогает пользователю быстрее понять, что

скрывается под той или иной панелью.

Для реализации вкладок применяется класс TabLayout. Для определения поведения вкладок (ScrollableTabs, FixedTabs или автоматический) применяются параметры MODE_AUTO, MODE_FIXED и MODED_SCROLLABLE. Вкладки с автоматическим изменением размера (MODE_AUTO) ведут себя как MODE_FIXED с GRAVITY_CENTER, вкладки соответствуют ширине содержимого TabLayout. Фиксированные вкладки имеют одинаковую ширину в зависимости от самой широкой метки (названия) вкладки. Когда вкладки превышают ширину представления, вкладки с автоматическим изменением размера ведут себя как MODE_SCROLLABLE, позволяя использовать динамическое количество вкладок без необходимости реализации дополнительной логики макета.

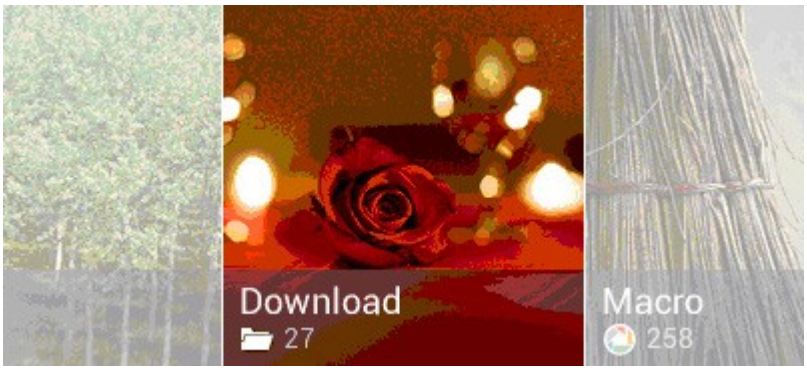

Рис. 70. Сетка с альбомами

МНОГООКОННЫЕ ПРИЛОЖЕНИЯ

Во второй части лабораторной работы рассмотрим способы разработки многоэкранных приложений. Предлагается разработать три приложения, каждое из которых посвящено отдельному аспекту этой тематики.

Пример 2. Создать многооконное приложение со списком

1. Создайте проект **MultiScreen**. В java-файле замените Activity на ListActivity. Класс ListActivity разработан таким образом, что на экране есть только прокручиваемый список и ему не нужна дополнительная разметка. Поэтому файл **activity_main.xml** можно удалить. Также следует удалить следующую строчку из класса MultiScreen:

```
setContentView(R.layout.activity_main);
```

т.к. layout-файл мы только что удалили.

2. Теперь нам нужен посредник, который свяжет список и названия элементов этого списка. Для этого в Android есть интерфейс Adapter. Нам потребуется наследник этого класса ArrayAdapter:

```
new ArrayAdapter(Context context, int textViewResourceId, String[] objects)
```

В качестве аргументов ArrayAdapter потребует текущий контекст, идентификатор ресурса с разметкой для каждой строки, массив строк.

Мы можем указать в качестве текущего контекста ListActivity (можно использовать ключевое слово this), готовый системный идентификатор ресурса (android.R.layout.simple_list_item_1) и созданный массив строк (String[] islands = { "Канары", "Курилы", "Мальдивы", "Филиппины"};). А выглядеть это будет так:

```
private ArrayAdapter<String> adapter;
adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, islands);
```

Обратите внимание на строчку android.R.layout.simple_list_item_1. В ней уже содержится необходимая разметка для элементов списка. Если вас не устраивает системная разметка, то можете создать собственную разметку в xml-файле и подключить её.

Осталось только подключить адаптер:

```
setListAdapter(adapter);
```

3. Теперь нам нужно подключить обработку нажатия. Для этого необходимо знать, на какой пункт списка осуществляется нажатие. Существует специальный интерфейс OnItemClickListener с методом onItemClick(). Подключаем обработчик: getListView().setOnItemClickListener(itemListener); Набираем все в том же onCreate

```
OnItemClickListener itemListener = new OnItemClickListener(){
}
```

Если подчеркнётся первое слово, то импортируем нужный класс. Далее подведём курсор ко второму слову new OnItemClickListener. Нам будет предложено добавить метод (Add unimplemented method). Соглашаемся и получаем заготовку:

```
OnItemClickListener itemListener = new OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                        long arg3) {
                // TODO Auto-generated method stub
```

```
        }
};
```

Меняем имена переменных arg на более привычные и понятные

```
public void onItemClick(AdapterView<?> parent,
View v, int position, long id);
```

4. Осталось описать, что будет происходить при нажатии на элемент.

По нашей задумке каждый элемент будет открывать новое окно с соответствующим содержимым.

Для начала следует создать еще 4 класса: Canari, Curili, Maldivi, Philippini, и 4 xml-оболочки к ним. Можно просто создать и копировать по одному файлу в обеих директориях, меняя только название и содержимое.

Например, создадим файлы **Canari.java** и **canari.xml** типа Activity. Обратите внимание, что как только мы создали ещё один java-файл, нужно записать его в файл манифеста, иначе приложение не будет видеть этот новый класс. Файл **AndtoidManifest.xml** находится сразу под папкой **res**. Добавьте код с именем класса в тегах <activity></activity>, сразу под главной активностью.

```
<activity       android:name="com.mypackage.multiscreen.Canari"
     android:label="@string/can" >
</activity>
```

Строку can нужно создавать в файле **strings.xml**, также как и другие строки.

Перейдем в файл **canari.xml** и создадим на экране TextView. Экраны будут отличаться друг от друга картинками. Возьмите 4 любые картинки с вашего компьютера (можете также найти их в интернете) и перетащите из проводника Windows в папку **res/drawable**. Теперь вы можете поместить элемент ImageView на экран и, выбрав нужную картинку из ресурсов проекта, подключить ее.

Остальные экраны создаются аналогично.

5. Теперь перейдем в главный класс и опишем обработку события onItemClick(). Создадим переключатель, который зависит от номера элемента.

```
switch (position) {
        case 0:
```

```
            break;
        case 1:

            break;
        case 2:

            break;

        case 3:

            break;
}
```

Для запуска нового экрана необходимо создать экземпляр класса Intent и указать класс, на который будем переходить (у нас их 4, поэтому для каждого случая выбираем свою). После этого вызывается метод startActivity(), который и запускает новый экран.

```
Intent intent = new Intent(MultiScreen.MainActivity.this, Canari.class);
                    startActivity(intent);
```

3. Теперь осталось добавить всплывающее окно **Toast**, которое будет показывать, какой элемент мы выбрали. Этот виджет можно импортировать так же, как и предыдущие. Нам потребуется метод makeText(), у которого есть три параметра: контекст приложения, текстовое сообщение и продолжительность времени показа уведомления.

```
Toast.makeText(getApplicationContext(), "Вы выбрали " +
parent.getItemAtPosition(position).toString(),Toast.LENGTH_SHORT).show();
```

Полные листинги файлов проекта, в которых были сделаны изменения, см. ниже.

```
package com.mypackage.multiscreen;

import android.os.Bundle;
import android.view.View;
import android.app.ListActivity;
import android.content.Intent;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Toast;

public class MultiScreenMainActivity extends ListActivity{

     String[] islands = { "Канары", "Курилы", "Мальдивы", "Филиппины"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);


      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, islands);
      setListAdapter(adapter);

      OnItemClickListener itemListener = new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {

                switch (position) {
                case 0:
                        Intent intent = new Intent(MultiScreenMainActivity.this,
Canari.class);
                        startActivity(intent);
                    break;
                case 1:
                        Intent intent1 = new
Intent(MultiScreenMainActivity.this, Curili.class);
                        startActivity(intent1);
                    break;
                case 2:
                        Intent intent2 = new
Intent(MultiScreenMainActivity.this, Maldivi.class);
                        startActivity(intent2);
                    break;

                case 3:
                        Intent intent3 = new
Intent(MultiScreenMainActivity.this, Philippini.class);
                        startActivity(intent3);
                    break;
                }
                Toast.makeText(getApplicationContext(), "Вы выбрали " +
parent.getItemAtPosition(position).toString(),
                        Toast.LENGTH_SHORT).show();
            }
        };
        getListView().setOnItemClickListener(itemListener);
    }
}
```

Листинг 7.1. Файл MultiScreenMainActivity.java

```
package com.mypackage.multiscreen;
import android.app.Activity;
import android.os.Bundle;

public class Canari extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.canari);
    }
}
```

Листинг 7.2. Файл Canari.java

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip" >

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="186dp"
        android:src="@drawable/canari" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="80dp"
        android:text="@string/enjoy"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Листинг 7.3. Файл canari.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Куда бы поехать в отпуск?</string>
    <string name="action_settings">Settings</string>
    <string name="enjoy">Enjoy yourself!</string>
    <string name="can">Канары</string>
    <string name="phil">Филиппины</string>
    <string name="cur">Курилы</string>
    <string name="mal">Мальдивы</string>
</resources>
```

Листинг 7.4. Файл strings.xml

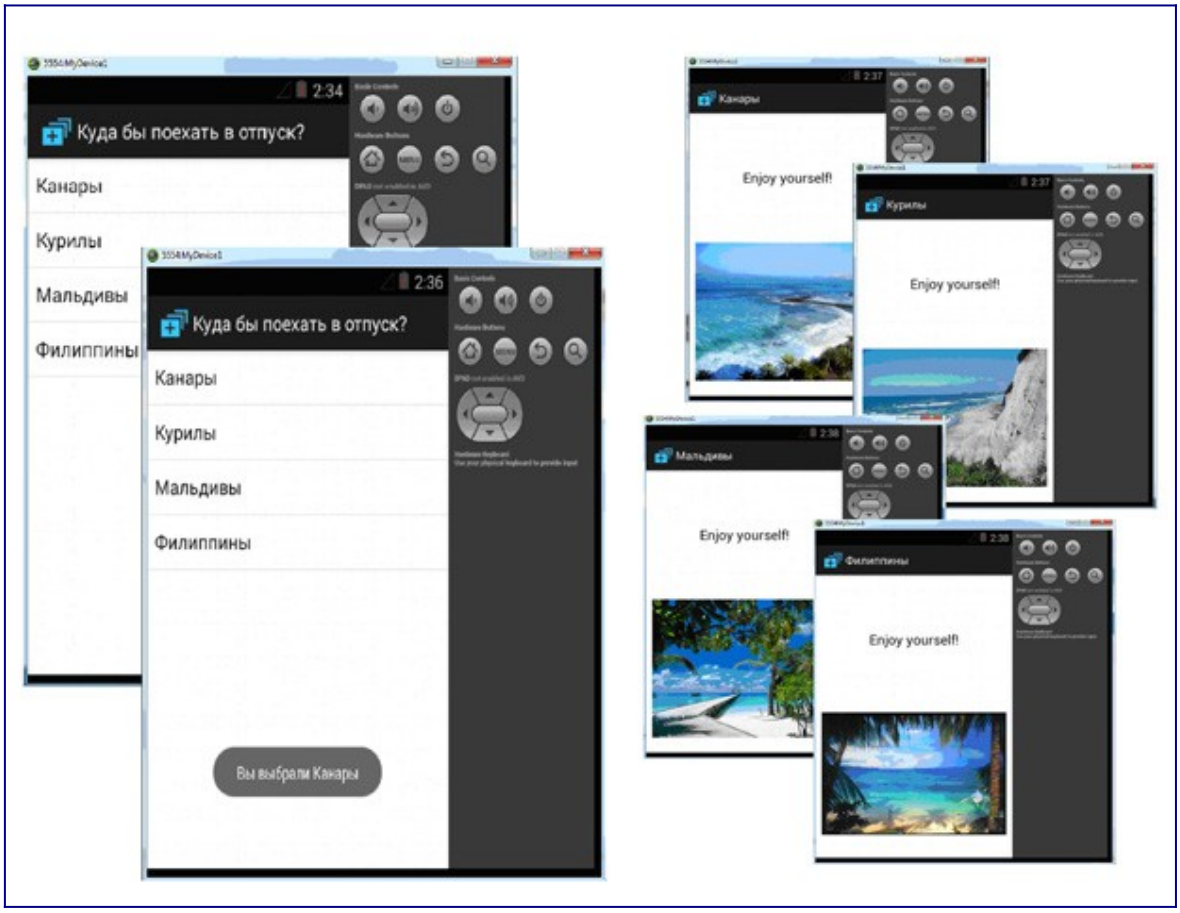

Рис. 71. Приложение "Куда бы поехать в отпуск?", запущенное на эмуляторе

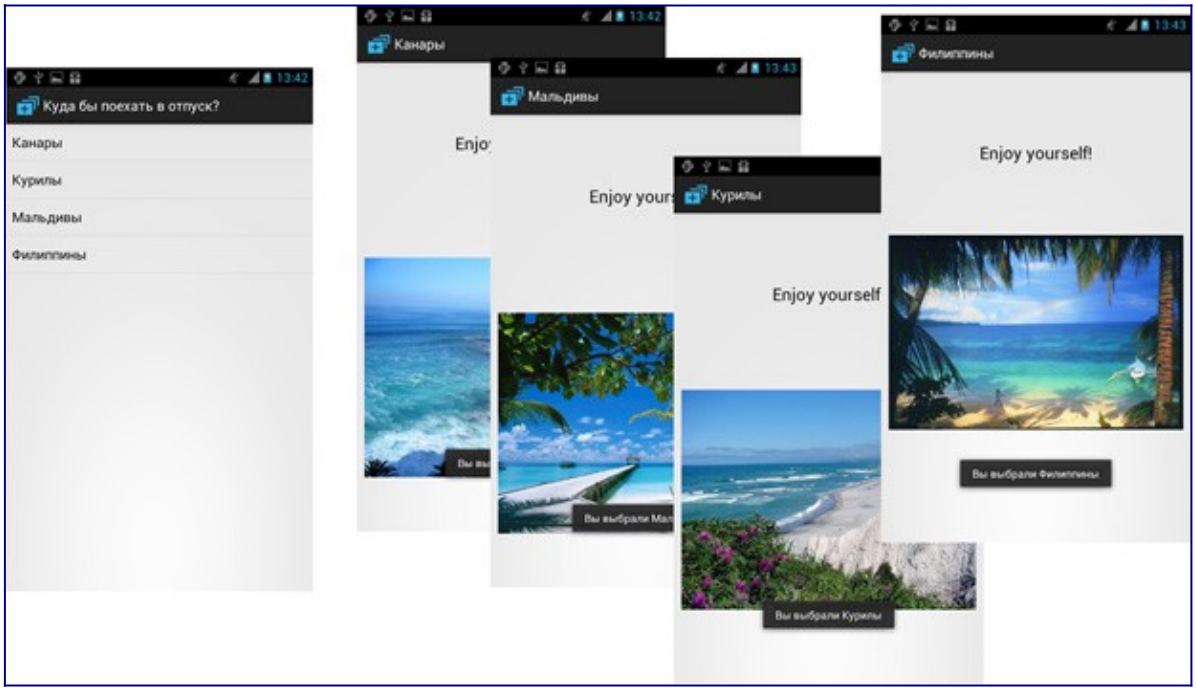

Рис. 72. Приложение "Куда бы поехать в отпуск?", запущенное на устройстве

Пример 3. Создать приложение с диалоговыми окнами

1. Создайте новый проект **Dialog**

2. Создайте кнопку на вашей активности и назовите ее "**Выбрать фон**" (для этого нужно в файле **strings** создать строку соответствующего содержания). Присвойте активности и кнопке **id**.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/relativelayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <Button
        android:id="@+id/background_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="174dp"
        android:text="@string/bg"/>
</RelativeLayout>
```

Листинг 5. Код файла activity_main.xml

3. Наше приложение меняет фон на выбранный. Значит, нам нужно создать цвета и их названия в файле **strings.xml**. Как вы помните, этот файл находится в папке **values**, которая в свою очередь находится в папке **res**. Также создадим строку messages, которая нам понадобятся для диалогового окна.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Dialog</string>
    <string name="action_settings">Settings</string>
        <string name="bg">Выбрать фон</string>
    <string name="message">Хотите поменять фон?</string>
    <color name="redColor">#FFFF0000</color>
    <color name="yellowColor">#FFFF00</color>
    <color name="greenColor">#FF00FF00</color>
    <string name="red">Красный</string>
    <string name="yellow">Жёлтый</string>
    <string name="green">Зелёный</string>
</resources>
```

Листинг 6. Файл strings.xml

4. Перейдем в файл **MainActivity.java**. Создайте следующие переменные:

```
private Button bgButton;
public RelativeLayout relativeLayout;
Context context;
```

Если компилятор подчеркивает тип и сообщает об ошибке, например, подчеркивает Context, наведите курсор на подчеркнутое слово: должно появиться контекстное меню, предлагающее варианты, как можно исправить ошибку. Выберете Import 'Context', чтобы импортировать библиотеку.

5. Теперь нужно описать, что будет происходить при нажатии на нашу кнопку.

Для начала свяжем объекты из **activity_main.xml** и переменные в **MainActivity.java** через **id** (в onCreate):

```
bgButton = (Button)findViewById(R.id.background_button);
relativeLayout = (RelativeLayout)findViewById(R.id.relativelayout);
```

Context - это объект, который предоставляет доступ к базовым функциям приложения.

Добавляем в код

```
context = MainActivity.this;
```

3. Теперь нужно добавить событие onClick и навесить на кнопку OnClickListener. Добавляем в заголовок класса MainActivity implements OnClickListener. Чтобы связать кнопку и Listener в onCreate пишем

```
bgButton.setOnClickListener(this);
```

Создадим теперь событие onClick

```
@Override
    public void onClick(View v){
}
```

В этом объекте создаем собственно диалог и называем его:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle(R.string.message);

AlertDialog alert = builder.create();
        alert.show();
```

Мы будем создавать диалоговое окно, предоставляющее пользователю выбор

из списка. Для этого потребуется ещё одна переменная, которая сформирует список из имеющихся строк.

```
final CharSequence[] items = {
getText(R.string.red), getText(R.string.yellow), getText(R.string.green)
};
```

7. Сформируем собственно наш список и зададим еще один Listener, который будет менять цвет фона на выбранный:

```
builder.setItems(items, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int item) {
              switch (item) {
              case 0: { relativeLayout.setBackgroundResource(R.color.redColor);
              break;}
              case 1:
{relativeLayout.setBackgroundResource(R.color.yellowColor);
              break;}
              case 2:
{relativeLayout.setBackgroundResource(R.color.greenColor);
              break;}
              }
```

3. Осталось добавить в каждый case всплывающие окна Toast, и приложение полностью готово!

```
Toast.makeText(context, R.string.green, Toast.LENGTH_LONG).show();

package com.mypackage.dialog;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

    private Button bgButton;
    public RelativeLayout relativeLayout;
    Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bgButton = (Button) findViewById(R.id.background_button);
        bgButton.setOnClickListener(this);
```

```
        context = MainActivity.this;
        relativeLayout = (RelativeLayout)findViewById(R.id.relativelayout);
    }

    @Override
    public void onClick(View v) {

        final CharSequence[] items = {getText(R.string.red)  ,
getText(R.string.yellow),getText(R.string.green)

        };

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle(R.string.message);
        builder.setItems(items, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int item) {
              switch (item) {
              case 0: { relativeLayout.setBackgroundResource(R.color.redColor);
              Toast.makeText(context, R.string.red, Toast.LENGTH_LONG).show();

              break;}
              case 1:
{relativeLayout.setBackgroundResource(R.color.yellowColor);
              Toast.makeText(context, R.string.yellow,
Toast.LENGTH_LONG).show();

              break;}
              case 2:
{relativeLayout.setBackgroundResource(R.color.greenColor);
              Toast.makeText(context, R.string.green, Toast.LENGTH_LONG).show();

              break;}
              }
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
}
```

Листинг 7. Код файла MainActivity.java

Скриншоты работающего приложения (см. на рисунках ниже):

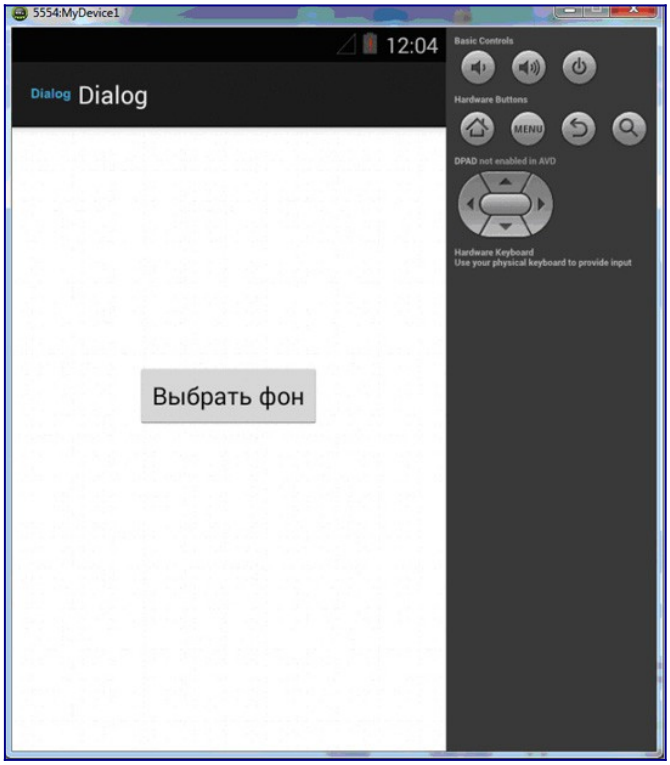

Рис. 73. Приложение "Dialog", запущенное на эмуляторе

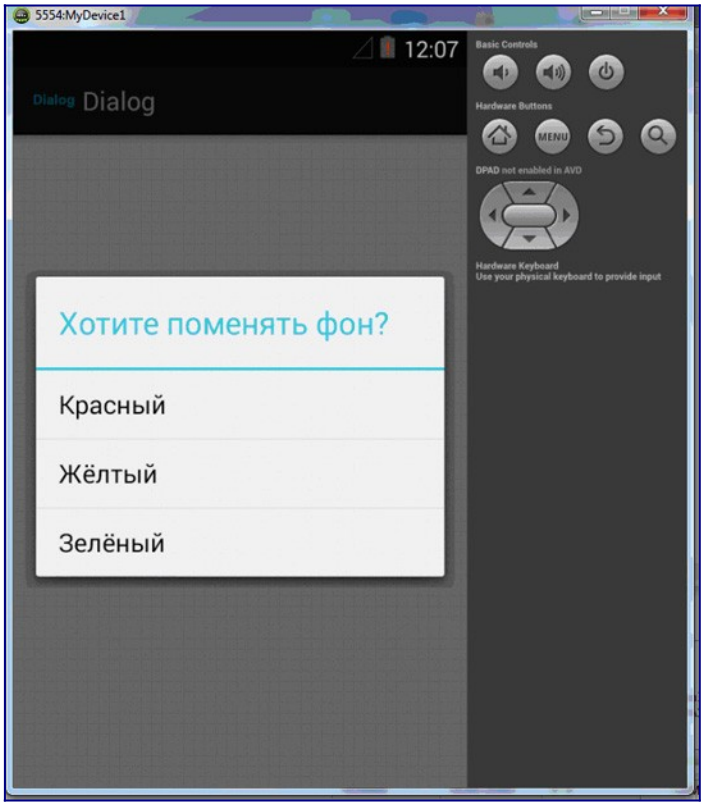

Рис. 73. Диалоговое окно

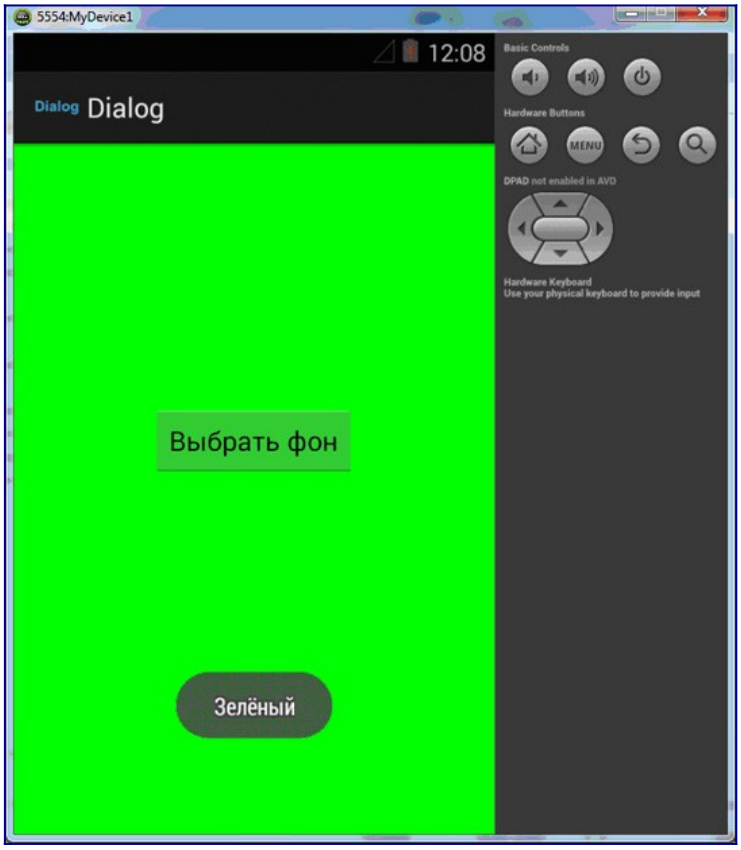

Рис. 74. Выбран зеленый фон

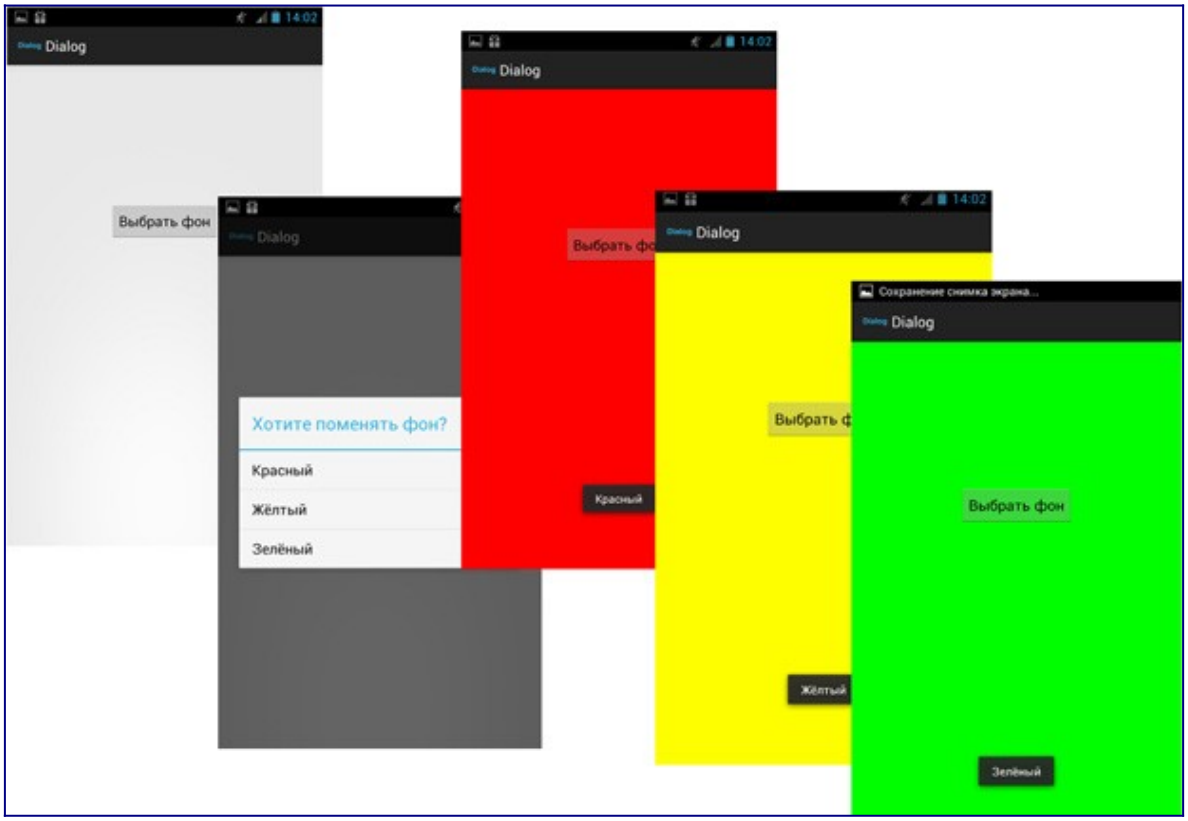

Рис. 75. Приложение "Dialog", запущенное на устройстве

Пример 4. Создать приложение со слайдингом из шаблона

1. Создайте проект **TabsAndSwipe**. Обратите внимание: чтобы использовать стандартный шаблон активности **Fixed Tabs + Swipe**, вам необходимо при создании проекта указать **Minimum Required SDK** не меньше, чем **API11**, т.к. в более ранних версиях этот шаблон не поддерживается.

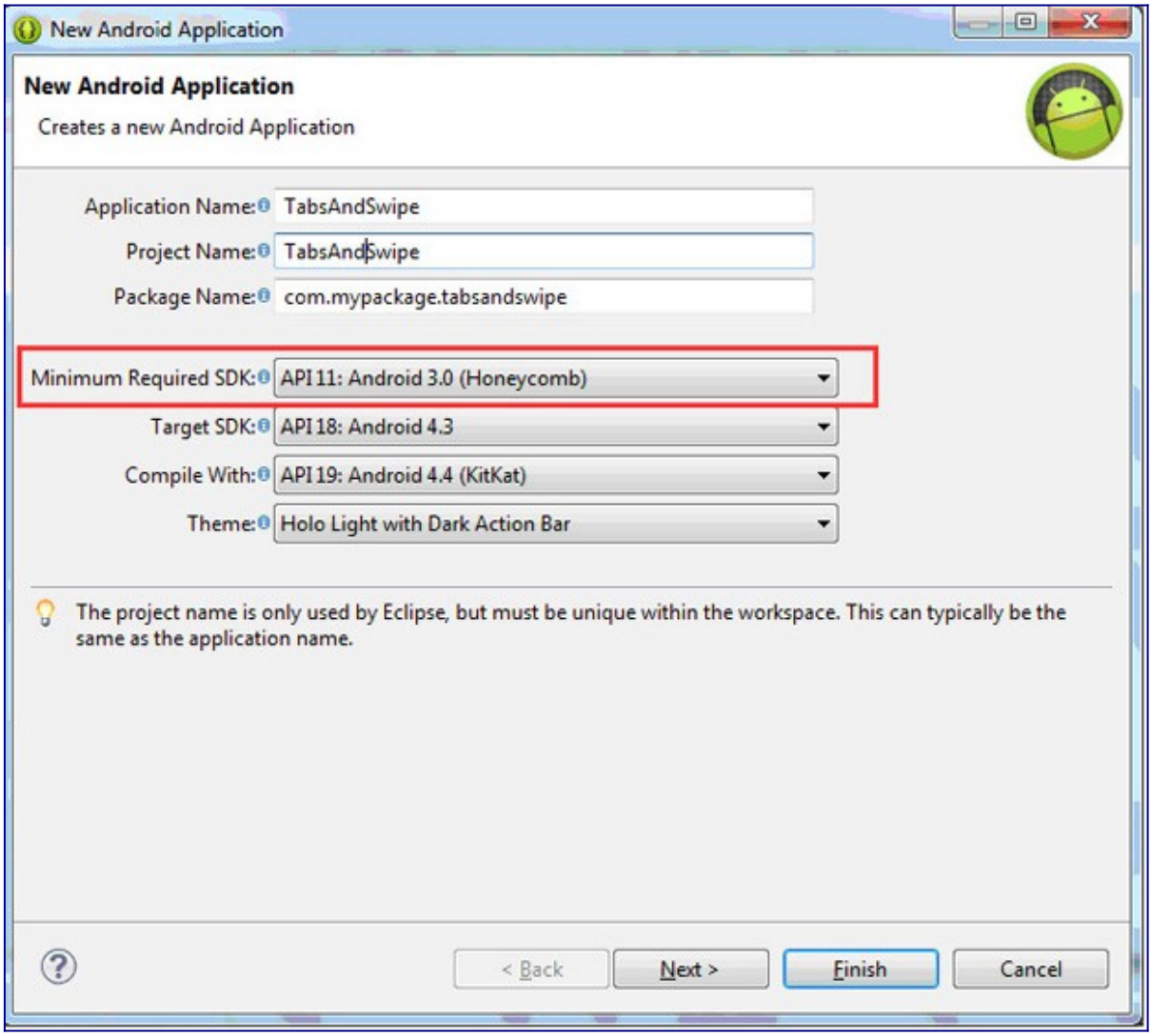

Рис. 76. Создание приложения TabsAndSwipe

Следующие три окна оставляем без изменений.

Далее в окне **Blank Activity** выбираем в графе **Navigation Type**.

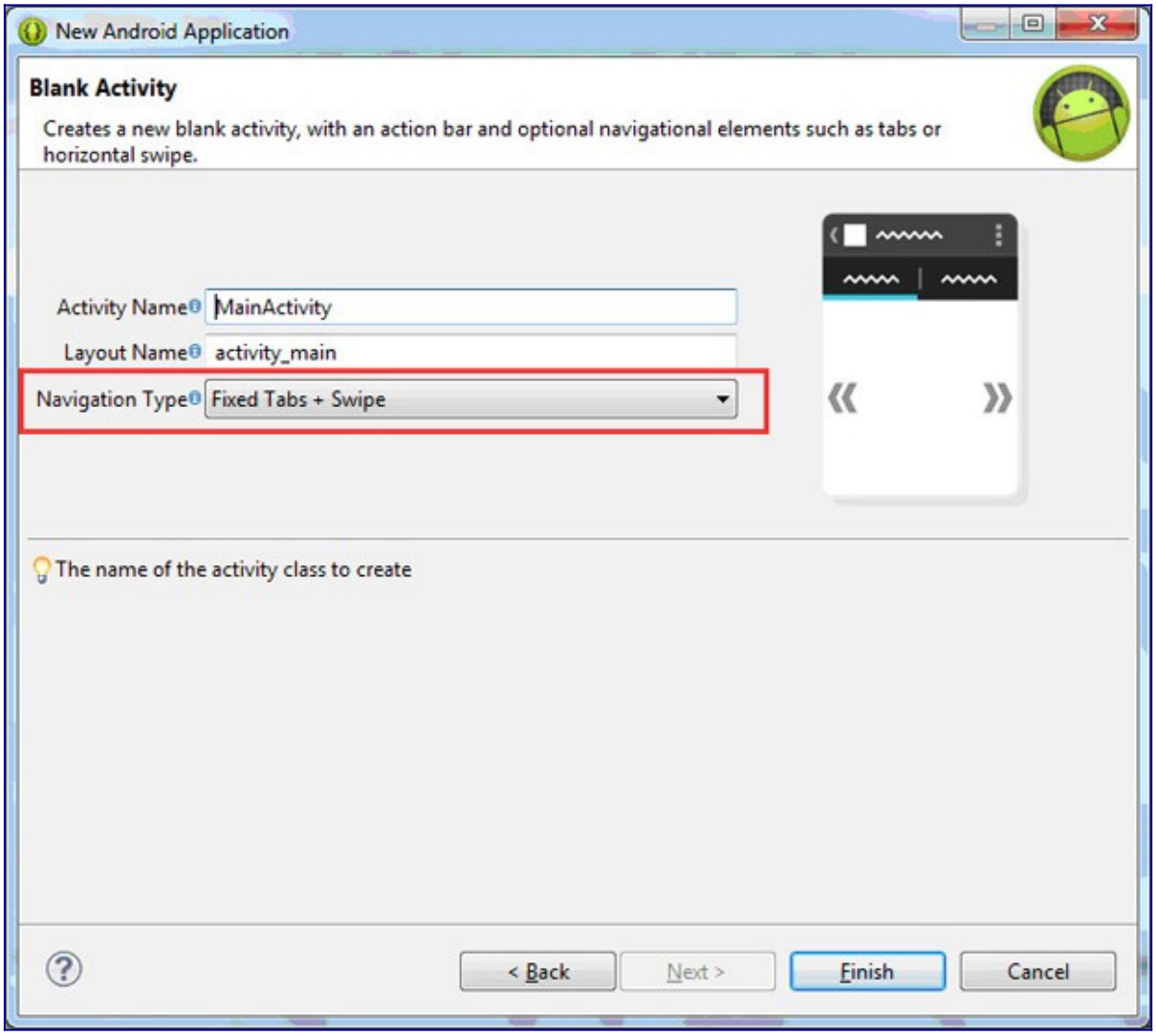

Рис. 77. Выбор шаблона Navigation Type

2. Посмотрите на структуру проекта: у вас появились два xml-файла в папке **res\layout - activity_main.xml** и **fragment_main_dummy.xml**.

3. Запустите приложение, чтобы убедиться, что все компилируется правильно.

4. Создайте три копии файла **fragment_main_dummy.xml**. Назовите их соответственно **first_tab**, **second_tab** и **third_tab**. Присвойте элементу TextView в каждом файле уникальный id. Можете поместить на экраны какие-набудь элементы, например картинки или надписи.

5. Теперь переходим к файлу **MainActivity.java**. Нас интересует класс

```
public static class DummySectionFragment extends Fragment
```

Делаем три копии данного класса со всем содержимым, называя их соответственно **FirstActivity**, **SecondActivity**, **ThirdActivity**.

3. Поменяйте в них следующие строки

```
View rootView = inflater.inflate(R.layout.fragment_main_dummy, container,
false);
TextView dummyTextView = (TextView) rootView.findViewById(R.id.section_label);
```

Замените R.layout.fragment_main_dummy на R.layout.first и R.id.section_label на R.id.section_label1 соответственно.

7. Поменяйте строки-названия секций в файле **strings.xml**.

```
<string name="title_section1">Лента</string>
    <string name="title_section2">Фото</string>
    <string name="title_section3">Карта</string>
```

3. Теперь переходим к классу SectionsPagerAdapter(). Нас интересует его метод Fragment getItem(), именно его мы будем изменять. Чтобы при перелистывании менялось не только содержимое TextView, но и всего фрагмента, замените код этого метода на следующий:

```
        public Fragment getItem(int position) {
            // getItem is called to instantiate the fragment for the given page.
            // Return a DummySectionFragment (defined as a static inner class
            // below) with the page number as its lone argument.
            Fragment fragment=null;
            Bundle args;
            switch (position) {
                case 0:
                    fragment = new FirstFragment();
                    args = new Bundle();
                    args.putInt(FirstFragment.ARG_SECTION_NUMBER, position + 1);
                    fragment.setArguments(args);
                    break;
                case 1:
                    fragment = new SecondFragment();
                    args = new Bundle();
                    args.putInt(SecondFragment.ARG_SECTION_NUMBER, position +
1);
                    fragment.setArguments(args);
                    break;
                case 2:
                    fragment = new ThirdFragment();
                    args = new Bundle();
                    args.putInt(ThirdFragment.ARG_SECTION_NUMBER, position + 1);
                    fragment.setArguments(args);
                    break;
            }
            return fragment;
        }
```

9. Приложение готово, можно запускать.

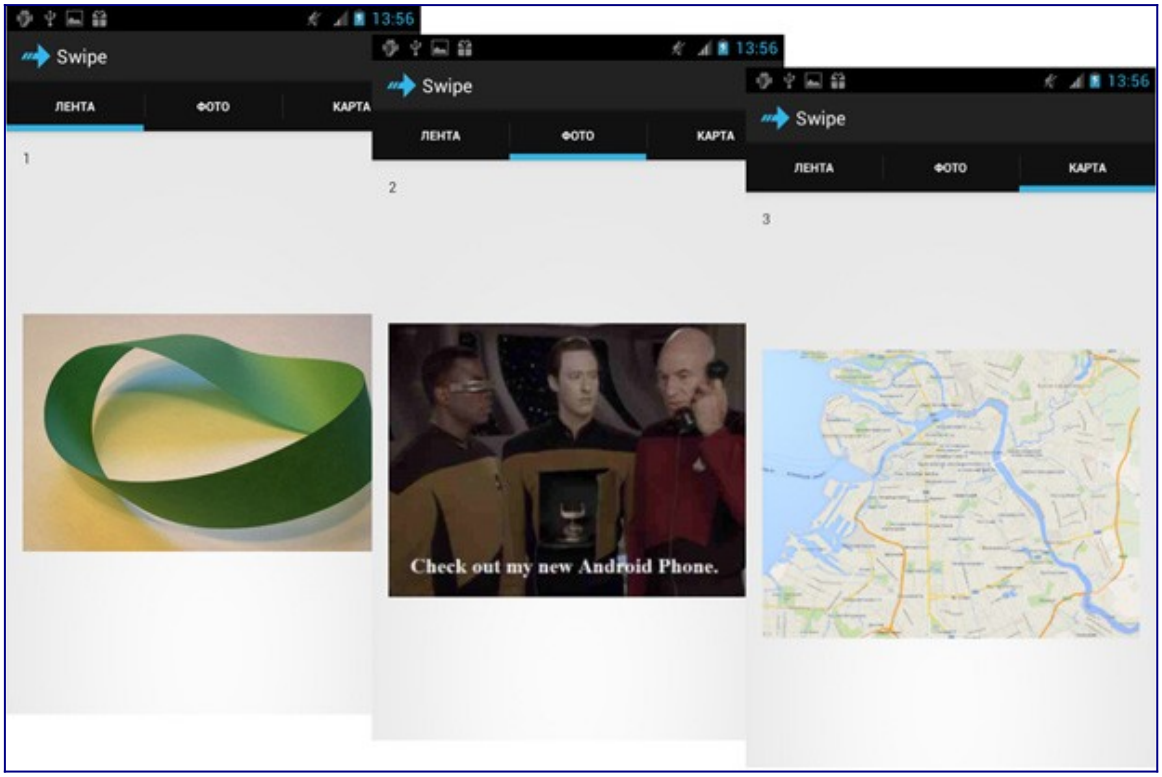

Рис. 78. Приложение TabsAndSwipe, запущенное на устройстве

РАБОТА С АНИМАЦИЕЙ

На экране расположены три кнопки (Frame animation, Transform animation, Cancel animation). При нажатии на первую кнопку воспроизводится покадровая анимация, при нажатии на вторую – анимация преобразований, при нажатии на третью анимация прекращается.

Импортируйте в рабочую область проект Animaton Example (см. ниже).

Запустите его на эмуляторе и протестируйте.

Ознакомьтесь с примерами создания покадровой анимации и анимации преобразований (res/anim/frame_anim.xml и res/anim/transform_anim.xml) и примерами их применения к объекту (src/MainActivity.java).

Пример 5. Анимация

res/anim/frame_anim.xml

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false" >
     <item
        android:drawable="@drawable/ic_launcher"
android:duration="200"/>
```

```
    <item
        android:drawable="@drawable/ic_launcher1"
android:duration="200"/>
    <item
        android:drawable="@drawable/ic_launcher2"
android:duration="200"/>
    <item
        android:drawable="@drawable/ic_launcher3"
        android:duration="200"/>

    </animation-list>
```

res/anim/transform_anim.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:shareInterpolator="false" >

    <scale
        android:duration="700"
android:fillAfter="false"
android:fromXScale="1.0"
android:fromYScale="1.0"

android:interpolator="@android:anim/accelerate_decelerate_interpolator"
android:pivotX="50%"          android:pivotY="50%"
android:toXScale="1.4"          android:toYScale="0.6" />

    <set android:interpolator="@android:anim/decelerate_interpolator" >
        <scale
            android:duration="400"
android:fillBefore="false"
android:fromXScale="1.4"
android:fromYScale="0.6"
android:pivotX="50%"
android:pivotY="50%"
android:startOffset="700"
android:toXScale="0.0"
android:toYScale="0.0" />

        <rotate
            android:duration="400"
android:fromDegrees="0"
android:pivotX="50%"
android:pivotY="50%"
android:startOffset="700"
android:toDegrees="-45"
android:toYScale="0.0" />
    </set>

    </set>
```

src/MainActivity.java

```
package com.example.application;
 import android.app.Activity;
import
android.graphics.Color;
```

```
import
android.graphics.drawable.AnimationDrawable;
import android.os.Bundle; import
android.view.View; import
android.view.View.OnClickListener; import
android.view.animation.Animation; import
android.view.animation.AnimationUtils; import
android.widget.Button; import
android.widget.ImageView;
 public class MainActivity extends Activity implements
OnClickListener {

 private Button startFrameAnim;
private Button startTransformAnim;
private Button cancelAnim;  private
ImageView animationView;

 @Override  public void onCreate(Bundle
savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  startFrameAnim = (Button) findViewById(R.id.frameAnimationStart);
  startTransformAnim= (Button) findViewById(R.id.transformAnimationStart);
  cancelAnim = (Button) findViewById(R.id.cancelAnimation);
  animationView = (ImageView) findViewById(R.id.animationView);

  startFrameAnim.setOnClickListener(this);
  startTransformAnim.setOnClickListener(this);
  cancelAnim.setOnClickListener(this);
 }
  public void onClick(View v) {
  if (startFrameAnim.equals(v)) {
           animationView.setBackgroundResource(R.anim.frame_anim);
 AnimationDrawable animation =
  (AnimationDrawable) animationView.getBackground();
            animation.start();
     } else if (startTransformAnim.equals(v)) {
          animationView.setBackgroundResource(R.drawable.ic_launcher);
 Animation transformAnimation =
  AnimationUtils.loadAnimation(this, R.anim.transform_anim);
  animationView.startAnimation(transformAnimation);
  } else if (cancelAnim.equals(v)) {
  animationView.setBackgroundColor(Color.BLACK);
       }
 }
}
```

Описание возможных элементов анимации преобразований представлено в таблице:

Элемент	Атрибуты
<alpha> анимация изменения прозрачности	**fromAlpha** – начальное значение прозрачности **toAlpha** – конечное значение прозрачности
<scale> анимация изменения размера	**fromxScale** – начальный масштаб по X **toxScale** – конечный масштаб по X **fromYScale** – начальный масштаб по Y **toYScale** – конечный масштаб по Y **pivotX** – X-координата закрепленного центра **pivotY** – Y-координата закрепленного центра
<translate> анимация движения (вертикальная/горизонтальная)	**fromXDelta** – начальное положение по X **toXDelta** – конечное положение по X **fromYDelta** – начальное положение по Y **toYDelta** – конечное положение по Y
<rotate> анимация вращения	**fromDegrees** – начальный угол вращения **toDegrees** – конечный угол вращения **pivotX** – координата X центра вращения **pivotY** – координата Y центра вращения

Задания для самостоятельной работы

ЗАДАНИЕ 1.

1. Внести изменения в приложение «Угадай число» из лабораторной работы 6, добавив следующий функционал:

- изменить верстку, проиллюстрировав использование макетов: RelativeLayout, FrameLayout, LinearLayout;
- использовать загружаемые изображения;
- использовать различные виды элементов интерфейса;
- вывод результата игры, используя диалоговое окно;
- использовать прогресс-бар для визуализации количества попыток;
- использовать анимацию изменения прозрачности, изменения размера, движения и вращения;
- использование комбинированных видов анимации, например изменение размера и движения.

ЗАДАНИЕ 2.

1. Внести изменения в приложение «Калькулятор» из лабораторной работы 6, добавив следующий функционал:

- реализовать в виде многооконного приложения;
- проиллюстрировав использование фрагментов;

- продемонстрировать использование различных элементов интерфейса от кнопок, чекбоксов, диалоговых окон, прогресс-бара и других.
- изменить верстку, проиллюстрировав использование макета RelativeLayout, FrameLayout, LinearLayout и других;
- применить анимационные эффекты.