Материальный дизайн. Меню. Навигация по приложению

Материальный дизайн. Menu, AppBar, ActionBar и ToolBar. Navigation Drawer. Темы, атрибуты тем, свои стили.

[Материальный дизайн](#_19lzj0y4r0r8)

[Единицы измерения и выравнивание](#_te70f5n9mte9)

[Использование тем материального дизайна](#_tol0vcehmr6)

[Меню в приложении](#_urmje8bovocb)

[Options menu и action bar](#_a90x45sw1p7g)

[Context menu and contextual action mode](#_1y1udlz3pppq)

[Pop-up menu](#_akbhumj3pp9l)

[Приложение action bar](#_238m7hg93jco)

[Определение меню в ресурсах (XML-файл)](#_lnxbz9)

[Добавление меню в action bar](#_35nkun2)

[Приложение context menu](#_4crqim1p3p0n)

[Application Bar](#_o5fuk9p4dibl)

[Навигация по приложению](#_p8tqop2ycly5)

[Другие виды дизайнов](#_mfp421bbd1l5)

[Стили и темы](#_2et92p0)

[Практическое задание](#_2s8eyo1)

[Дополнительные материалы](#_17dp8vu)

[Используемая литература](#_3rdcrjn)

# 

# Материальный дизайн

Материальный дизайн – это общая концепция для всего интерфейса системы. Он призван сделать цифровые объекты реалистичными, но не скевоморфными (визуально копирующими реальные). В таком дизайне пользователь работает со слоями приложения, или с «кусочками цифровой бумаги». Их называют тактильными поверхностями, и они обладают физическими свойствами.

Материальный дизайн описывает правила, которые определяют, каким образом элементы на экране должны двигаться и изменяться, взаимодействуя друг с другом. Главная идея материального дизайна – создать у пользователя ощущение, будто он работает с реальными объектами. Они должны двигаться естественно, чтобы пользователь понимал принципы и последовательность действий. Дизайн должен правильно организовывать пространство и движения элемента в нем. Объекты реагируют изменяющимися тенями и реакцией на касания.

Материальный дизайн основан на эффекте тактильной поверхности: пользователь может интуитивно почувствовать границы между поверхностями. Они подают сигналы и ориентируют пользователя. Должно создаваться ощущение, что одна поверхность располагается выше другой. Такой эффект создается благодаря границам и игре теней. Свет, тени, движения поверхностей – определяющие факторы для ориентировки пользователя. Освещение разделяет пространство и заостряет внимание на важных участках.

Графика, шрифт и цвет должны создать отчетливый дизайн, который определяет четкую иерархию элементов, понимание интерфейса. Компоновка элементов, акцентные цвета, четкие и понятные шрифты погружают пользователя в работу с приложением.

Действия пользователя (как инициатора приложения) первичны. Акцент ставится на основной функционал приложения, привлекающий внимание.

Все объекты должны перемещаться и изменяться синхронно, показывая, что происходит в каждый момент времени. Так интерфейс становится понятнее. Видимые объекты не должны нарушать дизайн даже после перемещения или преобразования. Движения обеспечивают смысл действий, привлекают внимание пользователя. Вместе с тем, движения не должны затягиваться.

## Единицы измерения и выравнивание

В материальном дизайне все элементы размещаются, выравниваясь по ячейкам сетки. Размер ячейки составляет 4 dp (четыре независимых от разрешения экрана пикселя). Расстояние от границы экрана до элемента обычно составляет 4 ячейки, или 16 dp. Это касается текстовых элементов. Отступ для изображений рекомендуется делать равным 72 dp.

## Использование тем материального дизайна

Можно использовать следующие возможности тем:

* Настройка цветовой палитры для изменения системных виджетов;
* Анимация системных виджетов при их касании;
* Анимация переходов между Activity.

В стандартной библиотеке находятся три темы:

* **@android:style/Theme.Material** (темная тема);
* **@android:style/Theme.Material.Light** (светлая тема);
* **@android:style/Theme.Material.Light.DarkActionBar** (светлая тема с ActionBar в темном стиле).

Более полную информацию про материальный дизайн можно найти на этих ресурсах:

* <https://awdee.ru/material-design-full-guide/>
* <https://material.io/>

# Меню в приложении

Меню – стандартный элемент интерфейса, часто используемый в приложениях. Чтобы все меню имели одинаковый шаблон и работали привычно для пользователей, Google рекомендует применять интерфейсы класса **android.view.Menu**.

Android предлагает 3 стандартных вида меню:

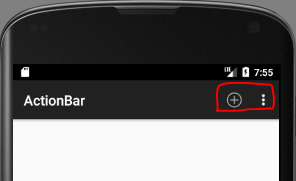
* **options menu** и **action bar**;
* **context menu** и **contextual action mode** (меню контекста);
* **pop-up menu** (всплывающее меню).

## Options menu и action bar

**Options menu** относится к конкретной **activity**, поэтому в нем должны находиться действия, относящиеся именно к ней («Поиск» или «Настройки»).

Пункты меню отображаются в **action bar**. Все кнопки по умолчанию размещаются в **overflow menu**. Это меню открывается нажатием на **action menu** в правой части **action bar**. Можно вынести несколько пунктов меню в **action bar** – обычно так поступают с пунктами меню, случайное нажатие которых не несет последствий (например, «Поиск»).

Меню назначается для activity и фрагментов. Когда несколько вложенных элементов определяют меню, они автоматически объединяются. Первыми будут пункты из activity, затем пункты фрагментов в том порядке, как они добавлялись в activity.



## Context menu and contextual action mode

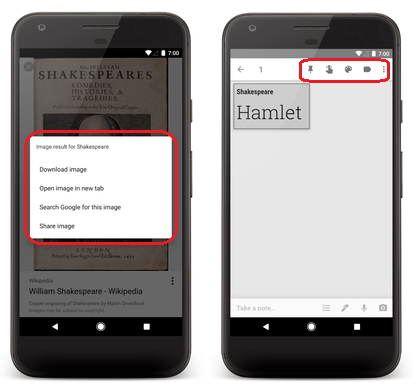
**Context menu** может определяться как для конкретного элемента, так и для любой области в UI. Чаще всего **context menu** используется для групп **view**, таких как **ListView** или **GridView**, где нужно выполнить действия для каждого элемента.

**ListView** является списком элементов, а **GridView** – это таблица элементов, которая тоже заполняется адаптером **ListAdapter**. При этом элементы располагаются в таблице. Одна из особенностей **GridView** в том, что он сам определяет количество столбцов. И разработчику не надо задумываться о расположении элементов. Наиболее часто GridView используется для отображения изображений, но есть и другие цели.

Рекомендуется разрабатывать подобные списки, применяя **RecyclerView** (там тоже возможно создавать списки и таблицы). Но для простого и статичного списка вполне применимы **ListView** или **GridView**.

Существует два вида **context menu**:

* **Floating context menu** – похоже на окно диалога, показывающее меню в виде списка. Появляется при длительном нажатии на **view**, к которому привязано. Действие всегда выполняется для одного выбранного элемента;
* **Contextual action mode** – приложение переходит в **action mode**, показывая при этом **action bar** с кнопками возможных действий. Строка находится вверху экрана, отображая кнопки, относящиеся к выбранным элементам. Здесь есть возможность одновременно выполнять действия с одним или несколькими выбранными элементами (в зависимости от реализации).

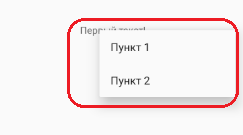
.

Google рекомендует пользоваться **contextual action mode** всегда, когда это возможно.

## Pop-up menu

**Pop-up menu** реализуется с помощью класса **PopupMenu** (доступно с уровня API от 11 и выше). **Pop-up menu** привязывается к **view** и отображается поверх него или ниже (если есть место). Сфера использования:

* Отображение пунктов меню, относящихся к выбранному элементу (выглядит примерно как меню, всплывающее при нажатии на правую кнопку мыши в Windows). В отличие от ранее описанных видов меню, контент элемента при этом не меняется;
* Добавление второй части командной последовательности (к примеру, кнопки «Добавить», которая открывает всплывающее меню с разными вариантами добавления);
* Выпадающее меню, похожее на **Spinner**, не сохраняющее выделение элемента.



**Pop-up** (всплывающее) меню и контекстное меню обрабатываются схожим образом. Но **pop-up** надо запускать самому:

|  |
| --- |
| PopupMenu popup = new PopupMenu(context, view);  popup.inflate(R.menu.main\_menu);  popup.show(); |

# Приложение action bar

### Определение меню в ресурсах (XML-файл)

Пункты меню определяются в файле формата XML, что позволяет отделить их от кода Activity. Благодаря этому меню можно загружать и использовать в различных Activity или фрагментах.

Чтобы создать меню, нужно добавить файл XML в проект, в папку **res/menu/**. Формат поддерживает следующие элементы (теги):

* <**menu**> – контейнер пунктов меню, содержащий один или несколько вложенных элементов <**item**> и <**group**>;
* <**item**> содержит пункт меню, определяющийся классом **MenuItem**. Также может содержать вложенное меню;
* <**group**> – необязательный и невидимый контейнер пунктов, разделяющий пункты меню по категориям.

Пример меню:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <menu xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto">  <item  android:id="@+id/menu\_add"  android:icon="@android:drawable/ic\_menu\_add"  app:showAsAction="always"  android:title="@string/menu\_add" />  <item  android:title="@string/menu\_clear"  android:id="@+id/menu\_clear" />  </menu> |

**main\_menu.xml**

Элемент <**item**> может иметь несколько атрибутов: **title** – название, **icon** – иконка (если она будет отображаться), **id** – идентификатор элемента и **showAsAction**, который указывает, когда и как он должен отображаться в строке действий.

Атрибут **showAsAction** может принимать различные параметры:

* **always** – всегда видно, но при нехватке места заголовок будет сокращен;
* **ifRoom** – виден только при наличии места в меню;
* **never** – не видно никогда (чтобы его увидеть, нужно открыть меню);
* **withText** – виден с заголовком;
* **collapseActionView** – можно навесить дополнительный элемент, который сворачивается в кнопку, а при нажатии разворачивается на всю ширину **action bar**.

Если описанная выше **support library** подключена успешно – скорее всего, вы используете **appCompatActivity** вместо обычной activity. В режиме совместимости в меню добавлены еще следующие атрибуты:

* **xmlns:app="http://schemas.android.com/apk/res-auto"**
* **app:showAsAction**

### Добавление меню в action bar

Чтобы добавить меню из XML-файла, необходимо переопределить метод **onCreateOptionsMenu()** в **ListActivity** следующим образом:

|  |
| --- |
| @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main\_menu, menu);  return true;  } |

**MainActivity.java**

Чтобы обрабатывать пункты меню, следует переопределить метод **onOptionsItemSelected()**:

|  |
| --- |
| @Override  public boolean onOptionsItemSelected(MenuItem item) {  switch (item.getItemId()) {  case R.id.menu\_add:  addElement();  return true;  case R.id.menu\_clear:  clearList();  return true;  case R.id.menu\_info:  info();  default:  return super.onOptionsItemSelected(item);  }  } |

Полный текст **MainActivity.java**:

|  |
| --- |
| package ru.geekbrains.actionbar;  import android.app.Activity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.Toast;  public class MainActivity extends Activity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  }  // Меню Action bar – установка меню в action bar  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main\_menu, menu);  return true;  }  // Меню Action bar – выбор пункта меню  @Override  public boolean onOptionsItemSelected(MenuItem item) {  switch (item.getItemId()) {  case R.id.menu\_add:  addElement();  return true;  case R.id.menu\_clear:  clearList();  return true;  case R.id.menu\_info:  info();  default:  return super.onOptionsItemSelected(item);  }  }  private void addElement() {  Toast.makeText(this, "Добавление элемента", Toast.LENGTH\_SHORT).show();  }  private void clearList(){  Toast.makeText(this, "Очистка списка", Toast.LENGTH\_SHORT).show();  }  private void info(){  Toast.makeText(this, "Информация", Toast.LENGTH\_SHORT).show();  }  } |

**MainActivity.java**

Макет:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.constraint.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="ru.geekbrains.actionbar.MainActivity">  <TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Hello World!"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />  </android.support.constraint.ConstraintLayout> |

**layout\_main.xml**

Меню:

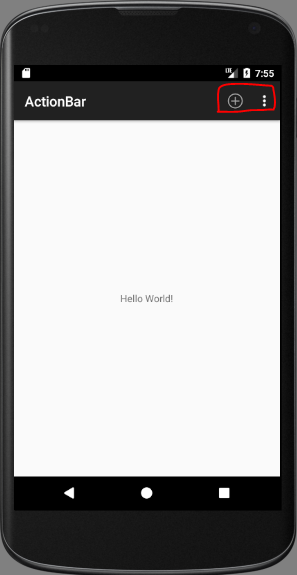
|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <menu xmlns:android="http://schemas.android.com/apk/res/android">  <item  android:id="@+id/menu\_add"  android:icon="@android:drawable/ic\_menu\_add"  android:showAsAction="always|withText"  android:title="@string/menu\_add" />  <item  android:id="@+id/menu\_info"  android:icon="@android:drawable/ic\_menu\_info\_details"  android:title="@string/menu\_info" />  <item  android:title="@string/menu\_clear"  android:id="@+id/menu\_clear" />  </menu> |

**menu\main\_menu.xml**

Ресурсы:

|  |
| --- |
| <resources>  <string name="app\_name">ActionBar</string>  <string name="menu\_add">Добавить</string>  <string name="menu\_clear">Очистить</string>  <string name="menu\_info">Информация</string>  </resources> |

**values\strings.xml**



<https://github.com/GeekBrainsTutorial/Android/tree/ActionBar/Android-2/Lesson-1/ActionBar>

# Приложение context menu

Для создания **floating context menu** регистрируем класс **View**, с которым связываем **context menu**. Для этого вызываем метод **registerForContextMenu()** и передаем туда **View**:

|  |
| --- |
| registerForContextMenu(view); |

Далее реализуем метод **onCreateContextMenu()**, который будет вызываться каждый раз перед тем, как будет показано контекстное меню:

|  |
| --- |
| @Override  public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {  super.onCreateContextMenu(menu, v, menuInfo);  MenuInflater inflater = getMenuInflater();  inflater.inflate(R.menu.context\_menu, menu);  } |

Для обработки касаний кнопок меню необходимо реализовать метод **onContextItemSelected()**:

|  |
| --- |
| @Override  public boolean onContextItemSelected(MenuItem item) {  switch (item.getItemId()) {  case R.id.menu\_item1:  Toast.makeText(this, "Выбран Пункт 1", Toast.LENGTH\_SHORT).show();  return true;  case R.id.menu\_item2:  Toast.makeText(this, "Выбран Пункт 2", Toast.LENGTH\_SHORT).show();  return true;  default:  return super.onContextItemSelected(item);  }  } |

Полный текст приложения:

|  |
| --- |
| package ru.geekbrains.contextmenu;  import android.os.Bundle;  import android.support.v7.app.AppCompatActivity;  import android.view.ContextMenu;  import android.view.MenuInflater;  import android.view.MenuItem;  import android.view.View;  import android.widget.TextView;  import android.widget.Toast;  public class MainActivity extends AppCompatActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  TextView textView1 = findViewById(R.id.textView1);  TextView textView2 = findViewById(R.id.textView2);  registerForContextMenu(textView1);  registerForContextMenu(textView2);  }  // Метод вызывается каждый раз перед тем,  // как context-ное меню будет показано  @Override  public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {  super.onCreateContextMenu(menu, v, menuInfo);  MenuInflater inflater = getMenuInflater();  inflater.inflate(R.menu.context\_menu, menu);  }  // Метод вызывается по нажатию на пункт меню. В качестве аргумента приходит item меню  @Override  public boolean onContextItemSelected(MenuItem item) {  switch (item.getItemId()) {  case R.id.menu\_item1:  Toast.makeText(this, "Выбран Пункт 1", Toast.LENGTH\_SHORT).show();  return true;  case R.id.menu\_item2:  Toast.makeText(this, "Выбран Пункт 2", Toast.LENGTH\_SHORT).show();  return true;  default:  return super.onContextItemSelected(item);  }  }  } |

**MainActivity.java**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.constraint.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="ru.geekbrains.contextmenu.MainActivity">  <TextView  android:id="@+id/textView1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Первый текст!"  app:layout\_constraintBottom\_toTopOf="@id/textView2"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />  <TextView  android:id="@+id/textView2"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="8dp"  android:layout\_marginTop="8dp"  android:text="Второй текст!"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintTop\_toBottomOf="@id/textView1"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent" />  </android.support.constraint.ConstraintLayout> |

**activity\_main.xml**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <menu xmlns:android="http://schemas.android.com/apk/res/android">  <item  android:id="@+id/menu\_item1"  android:title="Пункт 1" />  <item  android:id="@+id/menu\_item2"  android:title="Пункт 2" />  </menu> |

**context\_menu.xml**

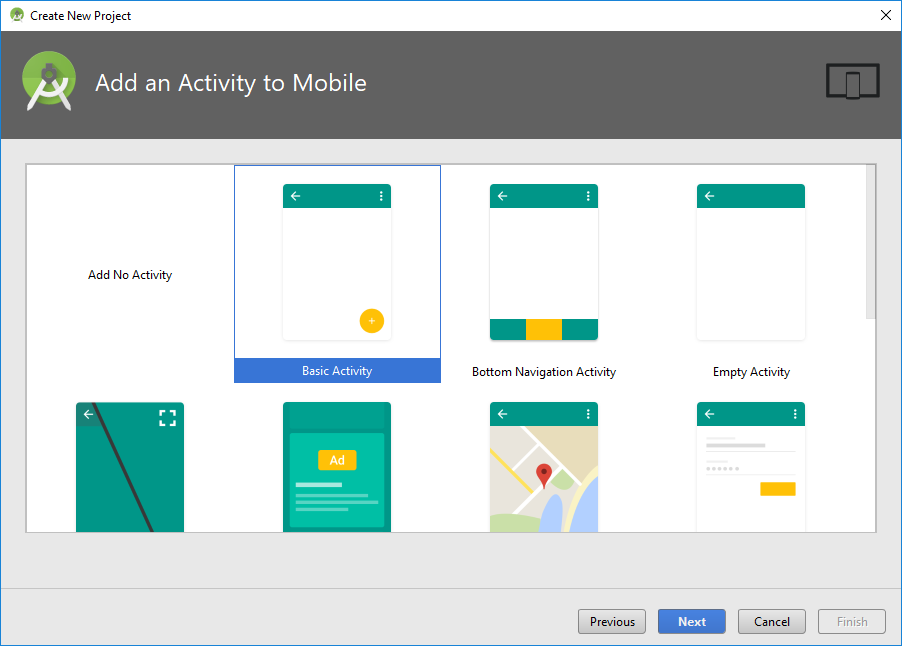


<https://github.com/GeekBrainsTutorial/Android/tree/ContextMenu/Android-2/Lesson-1/ContextMenu>

# 

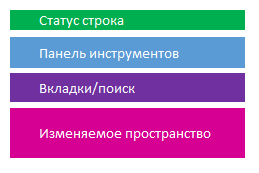
# Application Bar

**Application bar,** или **AppBar** – это верхняя часть экрана с расположенной там информацией, кнопками и меню. Фактически, элемент **AppBar** вырос из **action bar**. Теперь там появилась кнопка «Домой» и навигация по программе.

Для создания **AppBar** достаточно сделать проект на основе **Basic Activity**:

В макете можно обнаружить следующую конструкцию:

|  |
| --- |
| <android.support.design.widget.AppBarLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:theme="@style/AppTheme.AppBarOverlay">  <android.support.v7.widget.Toolbar  android:id="@+id/toolbar"  android:layout\_width="match\_parent"  android:layout\_height="?attr/actionBarSize"  android:background="?attr/colorPrimary"  app:popupTheme="@style/AppTheme.PopupOverlay" />  </android.support.design.widget.AppBarLayout> |

Мы видим элемент **AppBarLayout** – контейнер, который позволяет гибко изменять всю верхнюю часть экрана: заголовок, изображения, меню… Его можно настроить таким образом, чтобы при скроллинге изображение уменьшалось и пряталось, или вовсе исчезала вся панель.

**AppBar** включает в себя такие элементы, как:

* Статусная строка (status bar);
* Панель инструментов (tool bar);
* Панель вкладок или поиска (tab/search bar);
* Изменяемое пространство, рисунки и заголовки.

Статусная строка – это самая верхняя часть экрана, где показывается уровень сигнала и заряда аккумулятора, часы, значки запущенных приложений.

Заглянув в манифест, увидим, что в activity добавилась тема:

|  |
| --- |
| <activity  android:name=".MainActivity"  android:label="@string/app\_name"  android:theme="@style/AppTheme.NoActionBar"> |

В ней отключено создание **action bar**:

|  |
| --- |
| <style name="AppTheme.NoActionBar">  <item name="windowActionBar">false</item>  <item name="windowNoTitle">true</item>  </style> |

Теперь функции **action bar** на себя берет **tool bar** (панель инструментов), который вставляется в файл макета.

Чтобы включить панель инструментов (**tool bar**) в activity, надо выполнить следующие действия:

1. Задать **activity toolbar** из макета. Если используется пакет совместимости (support), то применяйте **setSupportActionBar()**; без пакета совместимости – **setActionBar()**:

|  |
| --- |
| Toolbar toolbar = findViewById(R.id.toolbar);  setSupportActionBar(toolbar); |

1. Переопределить метод **onCreateOptionsMenu()**, реализовав в нем меню:

|  |
| --- |
| public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.menu\_main, menu);  return true;  } |

1. Не забыть создать XML-файл меню:

|  |
| --- |
| <menu 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"  tools:context="ru.geekbrains.appbar.MainActivity">  <item  android:id="@+id/action\_info"  android:title="@string/info"  android:icon="@android:drawable/ic\_menu\_info\_details"  app:showAsAction="always"/>  <item  android:id="@+id/action\_settings"  android:orderInCategory="100"  android:title="@string/action\_settings"  app:showAsAction="never" />  </menu> |

**menu\main\_menu.xml**

После создания проекта изменился только файл activity:

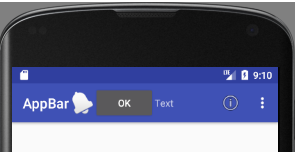
|  |
| --- |
| package ru.geekbrains.appbar;  import android.os.Bundle;  import android.support.design.widget.CoordinatorLayout;  import android.support.design.widget.FloatingActionButton;  import android.support.design.widget.Snackbar;  import android.support.v7.app.AppCompatActivity;  import android.support.v7.widget.Toolbar;  import android.view.View;  import android.view.Menu;  import android.view.MenuItem;  public class MainActivity extends AppCompatActivity {  private CoordinatorLayout mainLayout;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  mainLayout = findViewById(R.id.main\_layout);  Toolbar toolbar = findViewById(R.id.toolbar);  setSupportActionBar(toolbar);  FloatingActionButton fab = findViewById(R.id.fab);  fab.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View view) {  Snackbar.make(mainLayout, "Replace with your own action", Snackbar.LENGTH\_LONG)  .show();  }  });  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  // Добавляются пункты в Acrtion Bar, если они существуют  getMenuInflater().inflate(R.menu.menu\_main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  // Управление касаниями пунктов action bar  int id = item.getItemId();  // noinspection SimplifiableIfStatement  if (id == R.id.action\_settings) {  Snackbar.make(mainLayout, "Настройки", Snackbar.LENGTH\_LONG).show();  return true;  }  if(id == R.id.action\_info){  Snackbar.make(mainLayout, "Информация", Snackbar.LENGTH\_LONG).show();  return true;  }  return super.onOptionsItemSelected(item);  }  } |

**MainActivity.java**

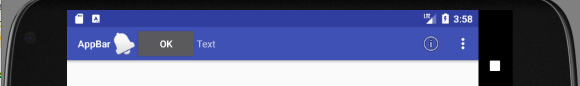
Теперь можно добавить в панель инструментов (tool bar) элементы. Макет:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.design.widget.CoordinatorLayout 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"  android:id="@+id/main\_layout"  tools:context="ru.geekbrains.appbar.MainActivity">  <android.support.design.widget.AppBarLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:theme="@style/AppTheme.AppBarOverlay">  <android.support.v7.widget.Toolbar  android:id="@+id/toolbar"  android:layout\_width="match\_parent"  android:layout\_height="?attr/actionBarSize"  android:background="?attr/colorPrimary"  app:popupTheme="@style/AppTheme.PopupOverlay">  <!-- Добавляем в tool bar элементы -->  <ImageView  android:src="@android:drawable/ic\_popup\_reminder"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content" />  <Button  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Ok"/>  <TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Text"/>  </android.support.v7.widget.Toolbar>  </android.support.design.widget.AppBarLayout>  <include layout="@layout/content\_main" />  <android.support.design.widget.FloatingActionButton  android:id="@+id/fab"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_gravity="bottom|end"  android:layout\_margin="@dimen/fab\_margin"  app:srcCompat="@android:drawable/ic\_dialog\_email" />  </android.support.design.widget.CoordinatorLayout> |

**activity\_main.xml**



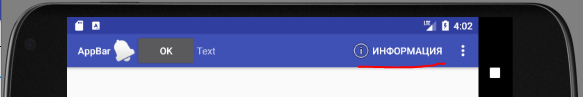
При повороте устройства в альбомную ориентацию видим такую картинку:



Чтобы текст пункта меню отобразился, надо задать свойство **withText**:

|  |
| --- |
| <item  android:id="@+id/action\_info"  android:title="@string/info"  android:icon="@android:drawable/ic\_menu\_info\_details"  app:showAsAction="always|withText"/> |

**main\_menu.xml**

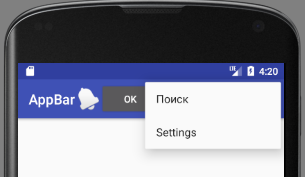


Видим текст пункта. При книжной ориентации текст пропадает, а на планшетах надпись будет появляться в любом положении.

Добавим еще один пункт меню:

|  |
| --- |
| <item  android:id="@+id/action\_search"  android:orderInCategory="10"  android:title="@string/action\_search" /> |

**main\_menu.xml**



Новый пункт находится выше предыдущего – это происходит независимо от того, в какое место его добавили в файле **main\_menu.xml**. Этим управляет свойство **android:orderInCategory="10"**. Рекомендуется задавать шаг между пунктами (например, через каждые 10), чтобы в дальнейшем было легче изменять порядок пунктов.

Изменим файл меню:

|  |
| --- |
| <menu 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"  tools:context="ru.geekbrains.appbar.MainActivity">  <item  android:id="@+id/action\_info"  android:title="@string/info"  android:icon="@android:drawable/ic\_menu\_info\_details"  app:showAsAction="always|withText"/>  <item  android:id="@+id/action\_settings"  android:orderInCategory="100"  android:title="@string/action\_settings"  app:showAsAction="never" />  <item  android:id="@+id/action\_search"  android:orderInCategory="10"  android:title="@string/action\_search" />  <group  android:checkableBehavior="all">  <item  android:id="@+id/choice"  android:title="Выбор"/>  </group>  <group  android:checkableBehavior="single">  <item  android:id="@+id/sort\_name"  android:title="По имени"/>  <item  android:id="@+id/sort\_date"  android:title="По дате"/>  </group>  </menu> |

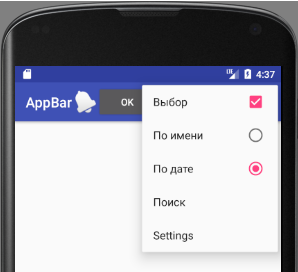
**main\_menu.xml**

Здесь добавились группы пунктов. В них можно добавлять свойства **android:checkableBehavior="single"** или **android:checkableBehavior="all"**. В первом случае мы получаем радиокнопки (**radiobutton**), а во втором – кнопки выбора (**checkbox**).

Чтобы показать выбор этих кнопок, надо добавить обработку в метод **onOptionsItemSelected()**:

|  |
| --- |
| @Override  public boolean onOptionsItemSelected(MenuItem item) {  // Handle action bar item clicks here. The action bar will  // automatically handle clicks on the Home/Up button, so long  // as you specify a parent activity in AndroidManifest.xml.  int id = item.getItemId();  // noinspection SimplifiableIfStatement  if (id == R.id.action\_settings) {  Snackbar.make(mainLayout, "Настройки", Snackbar.LENGTH\_LONG).show();  return true;  }  if(id == R.id.action\_info){  Snackbar.make(mainLayout, "Информация", Snackbar.LENGTH\_LONG).show();  return true;  }  // Добавим обработку radiobutton  if(id == R.id.sort\_date){  item.setChecked(true);  return true;  }  if(id == R.id.sort\_name){  item.setChecked(true);  return true;  }  // Добавим обработку checkbox  if(id == R.id.choice){  item.setChecked(!item.isChecked());  return true;  }  return super.onOptionsItemSelected(item);  } |

**MainActivity.java**



Чтобы реализовать поиск, необходимо в пункт меню поиска добавить следующие параметры:

|  |
| --- |
| <item  android:id="@+id/action\_search"  android:orderInCategory="10"  android:icon="@android:drawable/ic\_menu\_search"  app:showAsAction="ifRoom|collapseActionView"  app:actionViewClass="android.widget.SearchView"  android:title="@string/action\_search" /> |

**main\_menu.xml**

Добавим обработку поиска в activity: для этого надо изменить метод **onCreateOptionsMenu()**:

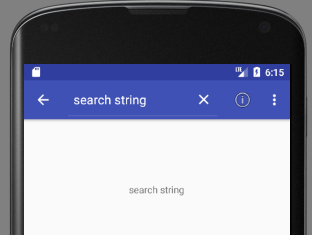
|  |
| --- |
| @Override  public boolean onCreateOptionsMenu(Menu menu) {  // Inflate the menu; this adds items to the action bar if it is present.  getMenuInflater().inflate(R.menu.menu\_main, menu);  MenuItem search = menu.findItem(R.id.action\_search); // Поиск пункта меню поиска  SearchView searchText = (SearchView) search.getActionView(); // Строка поиска  searchText.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  // Реагирует на конец ввода поиска  @Override  public boolean onQueryTextSubmit(String query) {  TextView s = findViewById(R.id.hello\_world);  s.setText(query);  return true;  }  // Реагирует на нажатие каждой клавиши  @Override  public boolean onQueryTextChange(String newText) {  return true;  }  });  return true;  } |

**MainActivity.java**

Чтобы было видно значение, введенное в строку поиска, задействуем существующий текстовый элемент. Исправляем макет:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.constraint.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"  app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  tools:context="ru.geekbrains.appbar.MainActivity"  tools:showIn="@layout/activity\_main">  <TextView  android:id="@+id/hello\_world"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Hello World!"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.186" />  </android.support.constraint.ConstraintLayout> |

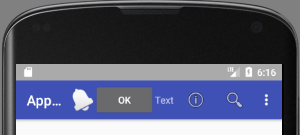
**content\_main.xml**



Чтобы поменять цвет статусной строки, достаточно исправить цвет темы в файле **colors.xml**:

|  |
| --- |
| <resources>  <color name="colorPrimary">#3F51B5</color>  <color name="colorPrimaryDark">#AAAAAA</color>  <color name="colorAccent">#FF4081</color>  </resources> |

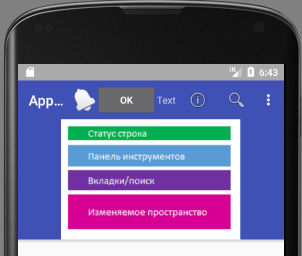
**colors.xml**



Можно добавить изображение **App bar**. Для этого надо добавить **ImageView** в **AppBarLayout**:

|  |
| --- |
| <ImageView  android:src="@drawable/image"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" /> |

Результат:

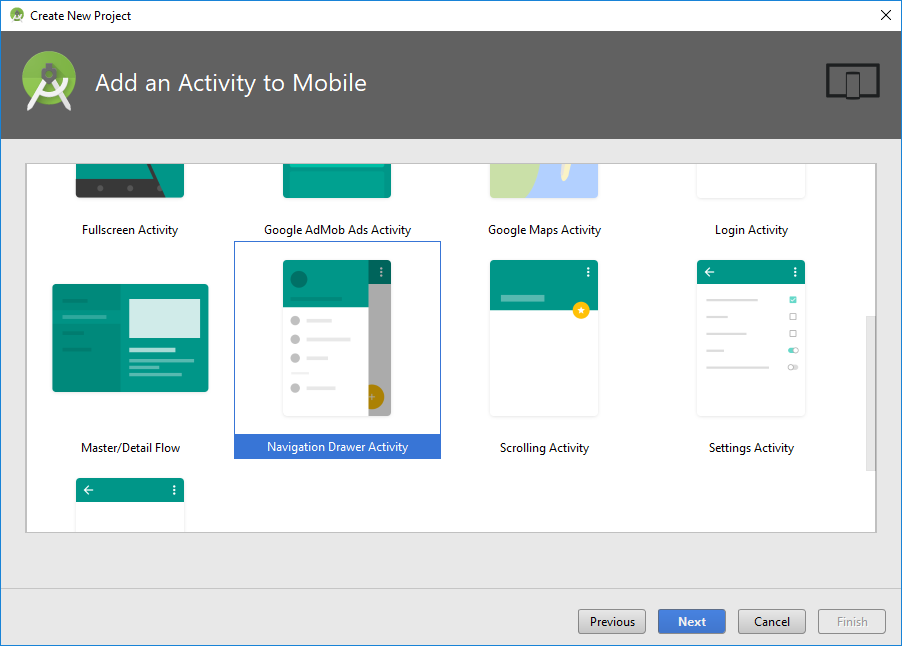


<https://github.com/GeekBrainsTutorial/Android/tree/AppBar/Android-2/Lesson-1/AppBar>

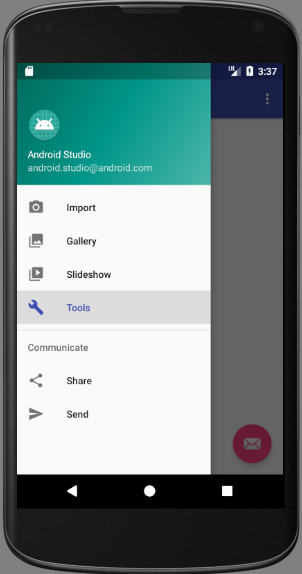
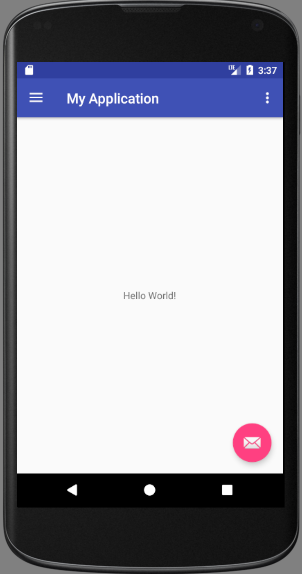
# 

# Навигация по приложению

На прошлых занятиях мы рассматривали тему навигации по приложению. Материальный дизайн упростил эту задачу. Выберем новый проект, как показано на рисунке:



Если теперь запустить приложение, получим следующую картинку:



Видим четыре макета. Два из них нам уже известны: это **app\_bar\_main.xml** и **content\_main.xml**. Они создаются такими же, как и на прошлом уроке (название у **app\_bar\_main.xml** изменилось, а содержание – нет).

Наибольший интерес представляет макет **activity\_main.xml**:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.v4.widget.DrawerLayout 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:id="@+id/drawer\_layout"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:fitsSystemWindows="true"  tools:openDrawer="start">  <include  layout="@layout/app\_bar\_main"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  <android.support.design.widget.NavigationView  android:id="@+id/nav\_view"  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  android:layout\_gravity="start"  android:fitsSystemWindows="true"  app:headerLayout="@layout/nav\_header\_main"  app:menu="@menu/activity\_main\_drawer" />  </android.support.v4.widget.DrawerLayout> |

**activity\_main.xml**

Основной контейнер – **DrawerLayout**, внутри него подключается макет **app\_bar\_main**. Макет должен быть первым элементом в **DrawerLayout**. Также есть еще один элемент: **NavigationView**.

Если мы рассмотрим его подробнее, увидим макет заголовка **app:headerLayout="@layout/nav\_header\_main"** и меню навигации **app:menu="@menu/activity\_main\_drawer".** Также в элементе **NavigationView** должно присутствовать свойство **android:layout\_gravity**. Как правило, этому свойству присваивается значение **Start** (слева – на языках, где пишут слева направо, справа – у пишущих справа налево).

В заголовок можно включить все, что угодно:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="match\_parent"  android:layout\_height="@dimen/nav\_header\_height"  android:background="@drawable/side\_nav\_bar"  android:gravity="bottom"  android:orientation="vertical"  android:paddingBottom="@dimen/activity\_vertical\_margin"  android:paddingLeft="@dimen/activity\_horizontal\_margin"  android:paddingRight="@dimen/activity\_horizontal\_margin"  android:paddingTop="@dimen/activity\_vertical\_margin"  android:theme="@style/ThemeOverlay.AppCompat.Dark">  <ImageView  android:id="@+id/imageView"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:paddingTop="@dimen/nav\_header\_vertical\_spacing"  app:srcCompat="@mipmap/ic\_launcher\_round" />  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:paddingTop="@dimen/nav\_header\_vertical\_spacing"  android:text="Android Studio"  android:textAppearance="@style/TextAppearance.AppCompat.Body1" />  <TextView  android:id="@+id/textView"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="android.studio@android.com" />  </LinearLayout> |

**nav\_header\_main.xml**

Здесь есть одно изображение и два текстовых поля. Кроме того, указан фон **android:background="@drawable/side\_nav\_bar"**. Фактически, это указание градиента.

В меню навигации перечислены пункты меню:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <menu xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  tools:showIn="navigation\_view">  <group android:checkableBehavior="single">  <item  android:id="@+id/nav\_camera"  android:icon="@drawable/ic\_menu\_camera"  android:title="Import" />  <item  android:id="@+id/nav\_gallery"  android:icon="@drawable/ic\_menu\_gallery"  android:title="Gallery" />  <item  android:id="@+id/nav\_slideshow"  android:icon="@drawable/ic\_menu\_slideshow"  android:title="Slideshow" />  <item  android:id="@+id/nav\_manage"  android:icon="@drawable/ic\_menu\_manage"  android:title="Tools" />  </group>  <item android:title="Communicate">  <menu>  <item  android:id="@+id/nav\_share"  android:icon="@drawable/ic\_menu\_share"  android:title="Share" />  <item  android:id="@+id/nav\_send"  android:icon="@drawable/ic\_menu\_send"  android:title="Send" />  </menu>  </item>  </menu> |

**activity\_main\_drawer.xml**

Что касается activity, то многие вещи нам уже знакомы по предыдущему проекту, за исключением обработки навигационного меню. Разберемся по порядку. В методе **onCreate()** надо создать слушатель, который обрабатывает выскакивание меню навигации. Это объект класса **ActionBarDrawerToggle**:

|  |
| --- |
| DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);  ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(  this, drawer, toolbar, R.string.navigation\_drawer\_open, R.string.navigation\_drawer\_close);  drawer.addDrawerListener(toggle);  toggle.syncState(); |

Разберем конструктор этого объекта:

* Activity, к которой он будет прикреплен – **this**;
* Класс **DrawerLayout** – **drawer**;
* **Toolbar** (**@nullable**, можно задать как **null**) – **toolbar**;
* Текст, описывающий открытие – **R.string.navigation\_drawer\_open**;
* Текст, описывающий закрытие – **R.string.navigation\_drawer\_close**.

В методе **onCreate()** также зарегистрируем слушателя, который умеет обрабатывать пункты меню навигации:

|  |
| --- |
| NavigationView navigationView = (NavigationView) findViewById(R.id.nav\_view);  navigationView.setNavigationItemSelectedListener(this); |

В данном случае сама activity будет обрабатывать эти пункты, так как она реализует интерфейс **NavigationView.OnNavigationItemSelectedListener**:

|  |
| --- |
| @SuppressWarnings("StatementWithEmptyBody")  @Override  public boolean onNavigationItemSelected(MenuItem item) {  // Handle navigation view item clicks here.  int id = item.getItemId();  if (id == R.id.nav\_camera) {  // Handle the camera action  } else if (id == R.id.nav\_gallery) {  } else if (id == R.id.nav\_slideshow) {  } else if (id == R.id.nav\_manage) {  } else if (id == R.id.nav\_share) {  } else if (id == R.id.nav\_send) {  }  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);  drawer.closeDrawer(GravityCompat.START);  return true;  } |

Достаточно вставить обработку. Кроме того, в activity переопределен метод касания кнопки «Назад»: если пользователь находится в меню навигации, то при касании этой кнопки оно закрывается:

|  |
| --- |
| @Override  public void onBackPressed() {  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);  if (drawer.isDrawerOpen(GravityCompat.START)) {  drawer.closeDrawer(GravityCompat.START);  } else {  super.onBackPressed();  }  } |

Добавим еще один **Layout nav\_gallery.xml** – для переключения на него при выборе пункта **nav\_gallery**:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout 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:id="@+id/nav\_gallery"  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"  app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  tools:context="ru.geekbrains.navdrawer.MainActivity"  tools:showIn="@layout/app\_bar\_main">  <TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Gallery opened" />  </RelativeLayout> |

**nav\_gallery.xml**

Изменим метод **onNavigationItemSelected()**, чтобы подгружался этот макет:

|  |
| --- |
| @SuppressWarnings("StatementWithEmptyBody")  @Override  public boolean onNavigationItemSelected(MenuItem item) {  // Handle navigation view item clicks here.  int id = item.getItemId();  int optionId = R.layout.content\_main;  if (id == R.id.nav\_camera) {  } else if (id == R.id.nav\_gallery) {  optionId = R.layout.nav\_gallery;  } else if (id == R.id.nav\_slideshow) {  } else if (id == R.id.nav\_manage) {  } else if (id == R.id.nav\_share) {  } else if (id == R.id.nav\_send) {  }  ViewGroup parent = (ViewGroup) findViewById(R.id.content);  parent.removeAllViews();  View newContent = getLayoutInflater().inflate(optionId, parent, false);  parent.addView(newContent);  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);  drawer.closeDrawer(GravityCompat.START);  return true;  } |

<https://github.com/GeekBrainsTutorial/Android/tree/NavDrawer/Android-2/Lesson-1/NavDrawer>

# Другие виды дизайнов

До появления материального дизайна существовало несколько типов дизайнов: скевоморфизм, минимализм и плоский дизайн.

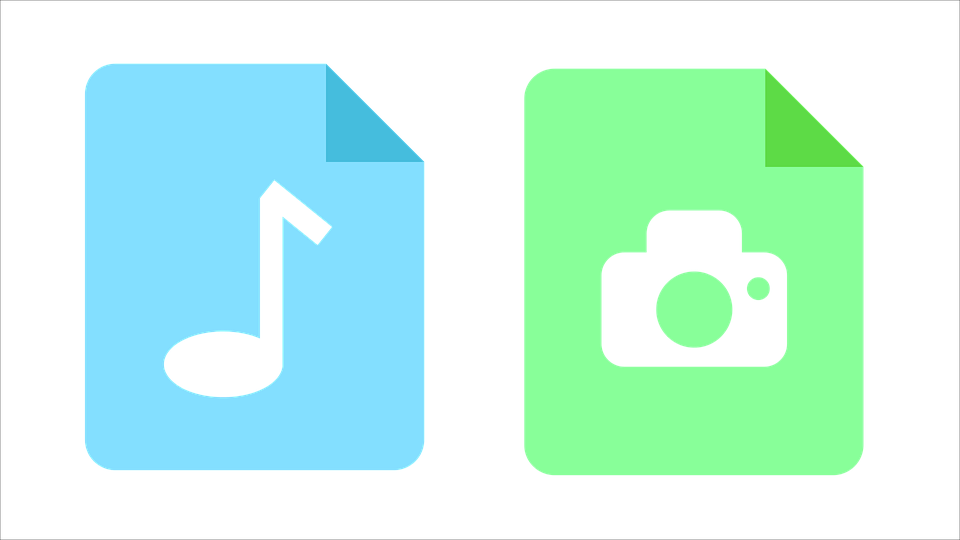
**Скевоморфизм** – это имитация текстуры реальных вещей. В программное обеспечение был принесен корпорацией Apple для того, чтобы обычные пользователи не пугались непонятных компьютерных значков и быстрее ориентировались. Дизайн цифрового калькулятора делал его изображение похожим на обычный калькулятор, а медиапроигрывателя – на телевизор.



**Плоский дизайн** пришел на смену скевоморфизму, когда пользователи освоились с компьютерными технологиями и обзавелись мобильной техникой. Этот дизайн работает быстрее, так как не тратит ресурсы на отрисовку сложных текстур.



**Минимализм**. Если в плоском дизайне элементы уже плоские, но все еще могут иметь градиентную раскраску, то в минимализме это недопустимо. Это наиболее простой дизайн.



**Дизайн Метро** был разработан на основе швейцарского дизайна, то есть на хорошей типографике и крупном тексте. Был взят на вооружении корпорацией Microsoft.



# Стили и темы

Стиль – это набор свойств, определяющих внешний вид. Это цвет, размер текста; цвет фона; высота, отступы и многое другое. Стиль определяется в файлах ресурсов отдельно от файла макета.

Стили в Android очень похожи на каскадные таблицы стилей в веб-дизайне. Здесь соблюдается принцип разделения дизайна и контента.

Стиль, применяемый к activity или приложению в целом, называется темой. Если применить стиль как тему, он применяется к каждому **view** в activity или приложении (если элементы имеют свойства, определяемые в теме).

Чтобы создать набор стилей, необходимо добавить XML-файл в папке **res/values/.** Название файла – на ваше усмотрение(\*.xml).

Элемент <**resources**> является корневым элементом. Сами стили описываются тэгом <**style**> с обязательным параметром **name**, значение которого должно быть уникальным.

Внутри каждого стиля может быть произвольное количество переопределяемых свойств. Свойство – это элемент <**item**> с параметром **name** и значением. **Name** задает полное название нужного параметра (например: "**android:layout\_width**" или "**android:textColor**").

Создадим стиль, описывающий шрифт, цвет и размер текста:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>  <style name="CodeFont" parent="@android:style/TextAppearance.Medium">  <item name="android:layout\_width">fill\_parent</item>  <item name="android:layout\_height">wrap\_content</item>  <item name="android:textColor">#00FF00</item>  <item name="android:typeface">monospace</item>  </style>  </resources> |

Применим этот стиль к **TextView**:

|  |
| --- |
| <TextView  style="@style/CodeFont"  android:text="@string/hello" /> |

Как видно из примера, атрибут **style** не имеет префикса **android**, а в остальном – ничего необычного.

Для удобства стили поддерживают наследование. Используя параметр **parent**, можно наследовать стиль. Например:

|  |
| --- |
| <style name="CodeFont" parent="@android:style/TextAppearance.Medium"> |

Создание темы выполняется идентичным образом (за тем исключением, что он назначается всей activity или приложению).

Создадим тему для приложения, унаследованную от стандартной:

|  |
| --- |
| <style name="CustomTheme" parent="android:Theme.Light">  <item name="android:windowBackground">@color/custom\_theme\_color</item>  <item name="android:colorBackground">@color/custom\_theme\_color</item>  </style> |

Чтобы применить ее к приложению, добавляем/изменяем параметр **android:theme** в тэг **application** файла **AndroidManifest.xml**:

|  |
| --- |
| <application android:theme="@style/CustomTheme"> |

Для применения темы к activity нужно таким же образом добавить параметр **android:theme** к нужной activity в файле **AndroidManifest.xml**:

|  |
| --- |
| <activity android:theme="@android:style/Theme.Dialog"> |

# Практическое задание

1. Создать **tool bar** для приложения с меню. Подумать, какие пункты меню вам нужны.
2. Добавить в проект **Navigation Drawer**. В **Drawer** добавить Header с аватаром пользователя, а также несколько пунктов: «О разработчике», «Форма обратной связи» + пункты на ваше усмотрение;
3. \* Добавить цветовую индикацию пунктов **Drawer**, чтобы всегда подсвечивался тот пункт, который выбран.

*Задание со “\*” – повышенной сложности*

# Дополнительные материалы

1. <https://developer.android.com/topic/libraries/support-library/index.html>
2. <https://developer.android.com/training/implementing-navigation/index.html>
3. <https://developer.android.com/guide/topics/ui/themes.html>

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. <https://stackoverflow.com/questions/26466877/how-to-create-context-menu-for-recyclerview>
2. <http://startandroid.ru/ru/uroki/vse-uroki-spiskom/177-urok-107-android-3-actionbar-razmeschenie-elementov.html>
3. <https://developer.android.com/training/appbar/index.html>
4. <https://material.io/guidelines/patterns/scrolling-techniques.html#>
5. [https://ru.stackoverflow.com/questions/498368/android-actionbar-background-image-не-работает](https://ru.stackoverflow.com/questions/498368/android-actionbar-background-image-%D0%BD%D0%B5-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%B5%D1%82)
6. <http://androiddocs.ru/toolbar-alternativa-actionbar/>
7. <https://www.sitepoint.com/better-user-interfaces-android-action-bar/>
8. <https://habrahabr.ru/post/222295/>
9. <https://appleinsider.ru/tag/skevomorfizm>
10. <https://ru.wikipedia.org/wiki/Metro_(%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)>
11. <https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D0%B8%D0%B7%D0%BC_(%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD)>
12. <https://coderwall.com/p/zpwrsg/add-search-function-to-list-view-in-android>
13. Пол Дейтел, Харви Дейтел, Александр Уолд. Android для разработчиков (3-е издание) – 2016 г.