7-ЛЕКЦИЯ. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

7.1. Использование списков (menus), виды(views) и рисунки (images)

Использование меню в приложениях позволяет сохранить ценное экранное пространство, которое в противном случае было бы занято относительно редко используемыми элементами пользовательского интерфейса.

Каждая Активность может иметь меню, реализующие специфические только для нее функции. Можно использовать также контекстные меню, индивидуальные для каждого Представления на экране.

В Android реализована поддержка трехступенчатой системы меню, оптимизированную, в первую очередь, для небольших экранов:

- Основное меню возникает внизу на экране при нажатии на кнопку «menu» устройства. Оно может отображать текст и иконки для ограниченного (по умолчанию, не больше шести) числа пунктов. Для этого меню рекомендуется использовать иконки с цветовой гаммой в виде оттенков серого с элементами рельефности. Это меню не может содержать радиокнопки и чекбоксы. Если число пунктов такого меню превышает максимально допустимое значение, в меню автоматически появляется пункт с надписью «еще» («more»). При нажатии на него отобразится Расширенное меню.
- Расширенное меню отображает прокручиваемый список, элементами которого являются пункты, не вошедшие в основное меню. В этом списке не могут отображаться иконки, но есть возможность отображения радиокнопок и чекбоксов. Поскольку не существует способа отобразить расширенное меню вместо основного, об изменении состояния каких-то компонентов приложения или системы рекомендуется уведомлять пользователя с помощью изменения иконок или текста пунктов меню.
- Дочернее меню (меню третьего уровня) может быть вызвано из основного или расширенного меню и отображается во всплывающем окне. Вложенность не поддерживается, и попытка вызвать из дочернего еще одно меню приведет к выбросу исключения.

При обращении к меню вызывается метод onCreateOptionsMenu Активности и для появления меню на экране его требуется переопределить. Данный метод получает в качестве параметра объект класса Menu, который в дальнейшем используется для манипуляций с пунктами меню.

Для добавления новых пунктов в меню используется метод add объекта Menu со следующими параметрами:

- Группа: для объединения пунктов меню для групповой обработки
- Идентификатор: уникальный идентификатор пункта меню. Этот идентификатор передается обработчику нажатия на пункт меню – методу onOptionsItemSelected.
- Порядок: значение, указывающее порядок, в котором пункты меню будут выводиться.
- Текст: надпись на данном пункте меню.

После успешного создания меню метод onCreateOptionsMenu должен вернуть значение true.

Пример показывает создание меню из трех пунктов с использованием строковых ресурсов:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, Menu.FIRST, Menu.NONE, R.string.menu_item1);
menu.add(0, Menu.FIRST+1, Menu.NONE, R.string.menu_item2);
menu.add(0, Menu.FIRST+2, Menu.NONE, R.string.menu_item3);
return true;
}
```

Для поиска пунктов меню по идентификатору можно использовать метод findItem объекта Menu.

Наиболее полезными параметрами пунктов меню являются следующие:

• Краткие заголовки: используются в случае, если пункт может отобразиться в основном меню. Устанавливается методом setTitleCondensed объекта класса MenuItem:

```
menuItem.setTitleCondensed("заголовок");
```

• Иконки: идентификатор Drawable, содержащий нужную картинку:

```
menuItem.setIcon(R.drawable.menu_item_icon);
```

• Обработчик выбора пункта меню: установить можно, но не рекомендуется из соображений повышения производительности, лучше использовать обработчик всего меню (onOptionsItemSelected). Тем не менее, пример:

```
menuItem.setOnMenuItemClickListener(new OnMenuItemClickListener() {
public boolean onMenuItemClick(MenuItem _menuItem) {
// обработать выбор пункта
return true;
}
});
```

• Намерение: это намерение автоматически передается методу startActivity, если нажатие на пункт меню не было обработано обработчиками onMenuItemClickListener и onOptionsItemSelected:

```
menuItem.setIntent(new Intent(this, MyOtherActivity.class));
```

Непосредственно перед выводом меню на экран вызывается метод onPrepareOptionsMenu текущей Активности, и переопределяя его можно динамически изменять состояние пунктов меню: разрешать/запрещать, делать невидимым, изменять текст и т. д.

Для поиска пункта меню, подлежащего модификации можно использовать метод findItem объекта Menu, передаваемого в качестве параметра:

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
super.onPrepareOptionsMenu(menu);
MenuItem menuItem = menu.findItem(MENU_ITEM);
//модифицировать пункт меню....
return true;
}
```

Android позволяет обрабатывать все пункты меню (выбор их) одним обработчиком onOptionsItemSelected. Выбранный пункт меню передается этому обработчику в качестве объекта класса MenuItem.

Для реализации нужной реакции на выбор пункта меню требуется определить, что именно было выбрано. Для этого используется метод getItemId переданного в качестве

параметра объекта, а полученный результат сравнивается с идентификаторами, использованными при добавлении пунктов в меню в методе onCreateOptionsMenu.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
// Проверить каждый известный пункт
case (MENU_ITEM):
// сделать что-то...
return true;
}
// Если пункт не обработан, отдаем обработку дальше
return super.onOptionsItemSelected(item);
}
```

При появлении на экране дочерние и контекстные меню выглядят одинаково, в виде плавающих окон, но при этом создаются по-разному.

Для создания дочерних меню используется метод addSubMenu объекта класса Menu:

```
SubMenu sub = menu.addSubMenu(0, 0, Menu.NONE, "дочернее меню");
sub.setHeaderIcon(R.drawable.icon);
sub.setIcon(R.drawable.icon);
MenuItem submenuItem = sub.add(0, 0, Menu.NONE, "пункт дочернего меню");
```

Как уже было сказано выше, вложенные дочерние меню Android не поддерживает.

Наиболее часто используемым способом создания контекстного меню в Android является переопределение метода onCreateContextMenu Активности:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
super.onCreateContextMenu(menu, v, menuInfo);
menu.setHeaderTitle("Конекстное меню");
menu.add(0, Menu.FIRST, Menu.NONE, "Пункт 1");
menu.add(0, Menu.FIRST+1, Menu.NONE, "Пункт 2");
menu.add(0, Menu.FIRST+2, Menu.NONE, "Пункт 3");
}
```

Регистрация обработчика контекстного меню для нужных Представлений осуществляется с помощью метода Активности registerForContextMenu:

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
tv = (TextView) findViewById(R.id.text_view);
registerForContextMenu(tv);
}
```

Пример обработчика:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
switch (item.getItemId()) {
case DELETE_ID:
AdapterContextMenuInfo info = (AdapterContextMenuInfo) item
.getMenuInfo();
```

```
db.deleteItem(info.id);
populate();
return true;
}
return super.onContextItemSelected(item);
}
```

Часто бывает удобнее всего описывать меню, в том числе иерархические, в виде ресурсов. О преимуществах такого подхода говорилось выше.

Меню традиционно описываются и хранятся в каталоге res/menu проекта. Все иерархии меню (если есть иерархические меню) должны находиться в отдельных файлах, а имя файла будет использовать как идентификатор ресурса. Корневым элементом файла должен быть тэг, а пункты меню описываются тэгом <item<. Свойства пунктов меню описываются соответствующими атрибутами:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
<item
android:id="@+id/item01"
android:icon="@drawable/menu_item"
android:title="Пункт 1">
</item>
<item
android:id="@+id/item02"
android:checkable="true"
android:title="Пункт 2">
</item>
<item
android:id="@+id/item03"
android:title="Пункт 3">
</item>
<item
android:id="@+id/item04"
android:title="Дочернее меню 1">
<menu>
<item
android:id="@+id/sub1item01"
android:title="Пункт дочернего меню 1">
</item>
</menu>
</item>
<item
android:id="@+id/item05"
android:title="Дочернее меню 2">
<menu>
<item
android:id="@+id/sub2item01"
android:title="Пункт дочернего меню 2">
</item>
</menu>
</item>
</menu>
```

Для создания объектов Menu из ресурсов в событий onCreateOptionsMenu и onCreateContextMenu используется метод inflate объекта типа MenuInflater:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.menu1, menu);
return true;
}
```

или так:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenuInfo menuInfo) {
super.onCreateContextMenu(menu, v, menuInfo);
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.menu2, menu);
menu.setHeaderTitle("Контекстное меню");
}
```

7.2. Галерея и ImageView

Элемент ImageButton – это особый тип кнопки, на которой вместо текста отображается изображение типа Drawable. На рис продемонстрирован элемент ImageButton, используемый для отображения изображения аватара, как часть экрана с настройками.

Оба элемента, ImageButton и Button, наследуются от класса View, однако больше они никак не связаны между собой. Класс Button – прямой потомок класса TextView (элемент Button можно рассматривать как строку текста, отображаемую поверх фонового изображения, выглядевшего как кнопка), а класс ImageButton – прямой потомок класса ImageView. Любое изображение, отображаемое при помощи элемента ImageButton, должно быть сохранено непосредственно на мобильном телефоне. Использование удаленных адресов типа Uri не рекомендуется, поскольку это может привести к снижению производительности и быстроты реакции приложения на действия пользователя.

Как и в случае с элементом ImageView, существует несколько различных вариантов установки изображения, отображаемого при помощи элемента ImageButton, включая следующие:

• setImageBitmap(). Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента ImageButton, существующий экземпляр класса Bitmap.

• setlImageDrawable(). Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента ImageButton, существующий экземпляр класса Drawable.

• setImageResource(). Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента ImageButton, существующий идентификатор ресурса.

• setlmageURI(). Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента ImageButton, существующий адрес типа Uri.

В некоторых ситуациях элемент ImageButton кэширует отображаемое изображение и продолжает делать это, даже если вы используете один из методов изменения графики. Один из вариантов решения этой проблемы - использование setImageURI(null), чтобы удалить кэшированную версию повторного вызова метода setImageURI() с аргументом Uri, содержащего адрес нового изображения, которое должно отображаться при помоги элемента ImageButton.

Ниже представлен удобный способ, позволяющий обращаться к ресурсам приложения, например к ресурсам типа Drawable, с использованием адреса типа Uri особого вида. Этот способ позволяет использовать метол setImageURI() элемента ImageButton как для графических ресурсов, так и для других изображений, хранящихся на мобильном телефоне. Обращаться к ресурсам при помощи адресов URI можно по идентификатору ресурса или по типу ресурса/его имени. Формат адреса типа Uri для варианта с использованием идентификатора ресурса выглядит следующим образом:android.resource://[пакет]/[идентификатор ресурса]

Например, вы могли бы использовать следующий адрес типа Uri для обращения к ресурсу типа Drawable с именем avatar.png по его идентификатору ресурса:

Формат адреса типа Uri для обращения к ресурсу по его типу/имени выглядит следующим образом:

android.resource://[пакет]/[тип ресурса]/[имя ресурса]

Например, вы могли бы использовать следующий адрес типа Uri для обращения к ресурсу типа Drawable с именем avatar.png по его типу/имени:

```
Uri path = Uri.parse(
“android.resource://com.androidbook.triviaquiz13/drawable/avatar”);
```

Когда у вас есть существующий адрес типа Uri для ресурса типа Drawable, вы можете использовать этот адрес для вызова метода setImageURI() элемента ImageButton, как показано в следующем коде:

```
ImageButton avatarButton= (ImageButton)
findViewById(R.id.ImageButton_Avatar);
avatarButton.setImageURI(path);
```

Для просмотра изображений предназначен элемент ImageView. С помощью свойства android:src можно задать отображаемое элементом изображение. Причем в качестве значения атрибута может выступать как файл изображения, так и ресурс drawable и даже обычный цвет:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

  <ImageView android:id="@+id/image1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/my_comps" />

  <ImageView android:id="@+id/image2"
    android:layout_width="125dip"
    android:layout_height="25dip"
     android:layout_marginTop="20dip"
     android:layout_marginBottom="20dip"
    android:src="#555555" />
```

```
    <ImageView android:id="@+id/image3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/mac_computer"
      android:scaleType="centerInside"
      android:maxWidth="30dip"
      android:maxHeight="30dip" />
</LinearLayout>
```

С помощью атрибута android:scaleType мы можем управлять положением изображения внутри ImageView. В данном случае изображение центрируется. Данный атрибут может принимать следующие значения:

• center: изображение центрируется по центру без масштабирования

• centerCrop: изображение центрируется по центру и масштабируется с сохранением аспектного отношения между шириной и высотой. Если какая-то часть не помещается в пределы экрана, то она обрезается

• centerInside: изображение центрируется по центру и масштабируется с сохранением аспектного отношения между шириной и высотой, но ширина и высота не могут быть больше ширины и высоты ImageView

• fitCenter: изображение масштабируется и центрируется

• fitStart: изображение масштабируется и устанавливается в начало элемента (вверх при портретной оринетации и влево - при альбомной)

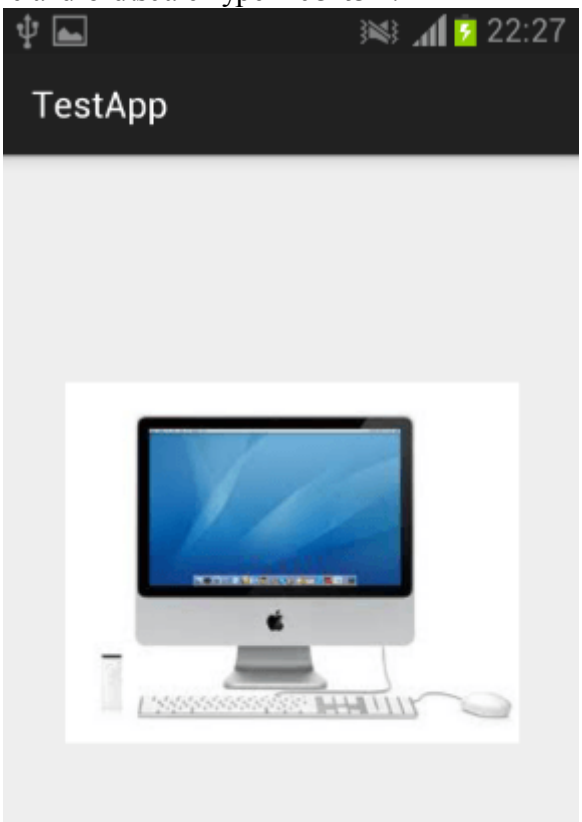
• fitEnd: изображение масштабируется и устанавливается в конец элемента (вниз при портретной оринетации и вправо - при альбомной)

• fitXY: изображение масштабируется без сохранения аспектного отношения межуд шириной и высотой, заполняя все пространство ImageView

• matrix: изображение масштабируется с применением матрицы изображения

Установка android:scaleType="fitXY":

Аналогичный пример с android:scaleType="center":



Используя набор методов можно присвоить ресурс изображения элементу в коде:
import android.widget.ImageView;

```
//.....

ImageView imgView = (ImageView)findViewById(R.id.image1);
imgView.setImageResource(R.drawable.images);
```

7.3. ImageSwitcher и GridView

Элемент ImageSwitcher позволяет деятельности анимировать переход между двумя элементами-представлениями ImageView. Вы добавили элемент ImageSwitcher с именем ImageSwitcher_QuestionText на макет игрового экрана для отображения изображения, связанного с каждым вопросом викторины, пользователю.

Для инициализации элемента ImageSwitcher вы просто указываете подходящую реализацию класса ViewFactory и затем используете один из трех методов, чтобы установить изображение. В следующем коду продемонстрировано использование метода setImageDrawable():

К сожалению, в настоящее время использовать метод setlmageURL() элемента ImageSwitcher для указания адресов URL ресурсов, размещенных в Интернете, невозможно. Вместо этого вам нужно проделать дополнительную работу, связанную с загрузкой изображения по указанному адресу URL и сохранением этого изображения в виде объекта типа Drawable. Представленная ниже реализация метода getQuestionImageDrawable() делает именно то, что нужно:

```
private Drawable getQuestionImageDrawable(int questionNumber) {
Drawable image;
URL imageUrl;
try {
// Create a Drawable by decoding a stream from a remote URL
imageUrl = new URL(getQuestionImageUrl(questionNumber));
Bitmap bitmap = BitmapFactory.decodeStream(imageUrl.openStream());
image = new BitmapDrawable(bitmap);
} catch (Exception e) {
Log.e(DEBUG_TAG, “Decoding Bitmap stream failed.”);
image = getResources().getDrawable(R.drawable.noquestion);
}
return image;
}
```

Метод getQuestionImageUrl() — это простой вспомогательный метод, который получает подходящий веб-адрес изображения для указанного вопроса. Эта информация хранится в объекте типа Hashtable с вопросами. (Мы поговорим о том, как обрабатываются вопросы, далее.) Полная реализация метода getQuestionlmageUrl() доступна в исходном коде на диске, прилагаемом к данной книге.

Класс URL используется для хранения удаленного адреса файла изображения в формате PNG, которое вы хотите загрузить в элемент ImageSwitcher. После этого вы выгружаете полученные данные в объект типа BitmapDrawable. Наконец, чтобы использовать методы для работы с потоком, приложению требуется разрешение android.permission.INTERNET, которое должно быть добавлено в файл манифеста Android.

Когда вы хотите обновить содержимое элемента ImageSwitcher, добавив новый элемент-представление ImageView, вы можете вызвать метод setImageDrawable():

```
ImageSwitcher questionImageSwitcher =
(ImageSwitcher) findViewById(R.id.ImageSwitcher_QuestionImage);
Drawable image = getQuestionImageDrawable(nextQuestionNumber);
questionImageSwitcher.setImageDrawable(image);
```

Вызов метода setImageDrawable() заставляет экземпляр класса MyImageSwitcher Factory сгенерировать новый элемент-представление irageView с объектом типа Drawable, переданным в качестве параметра метола setImageDrawable().

Чтобы анимировать переход между дочерними элементами-представлениями View элемента ViewSwitcher, применяются методы setInAnimation() и setOutAnimation(). Например, чтобы добавить анимацию плавного появления и плавного исчезновения элементов-предстаатений в элементе TextSwitcher, вы могли бы загрузить и использовать встроенные анимации платформы Android, как показано в следующем коде:

```
Animation in = AnimationUtils.loadAnimation(this, android.R.anim.fade_in);
Animation out = AnimationUtils.loadAnimation(this, android.R.anim.fade_out);
TextSwitcher questionTextSwitcher =
(TextSwitcher) findViewById(R.id.TextSwitcher_QuestionText);
questionTextSwitcher.setInAnimation(in);
questionTextSwitcher.setOutAnimation(out);
```

Теперь при каждом вызове метода setText() или метода setCurrentText() элемента TextSwitcher будет запускаться анимация, имитирующая плавное появление или исчезновение текста. Вы можете еще больше улучшить процесс перехода между вопросами, добавив предыдущие анимации к элементу ImageSwitcher, который используется для отображения изображений, связанных с вопросами.

Элемент GridView представляет отображение в виде таблицы - набора строк и столбцов. Создадим разметку GridView:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <GridView
        android:id="@+id/gridview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:columnWidth="100dp"
        android:numColumns="2"
        android:verticalSpacing="10dp"
        android:horizontalSpacing="10dp"
        android:stretchMode="columnWidth"
        android:gravity="center" />

</LinearLayout>
```

Теперь, как и в случае с ListView, надо установить связь с адаптером:

```
package com.example.eugene.myfirstapp;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
```

```
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.view.View;
import android.widget.GridView;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity{

    String[] countries = { "Бразилия", "Аргентина", "Чили", "Колумбия", "Уругвай",
"Парагвай"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_main);

      // получаем элемент ListView
      GridView countriesList = (GridView) findViewById(R.id.gridview);

      // создаем адаптер
      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, countries);
      countriesList.setAdapter(adapter);

      OnItemClickListener itemListener = new OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> parent, View v,
                    int position, long id) {
          Toast.makeText(getApplicationContext(),"Вы выбрали "
                  + parent.getItemAtPosition(position).toString(),
              Toast.LENGTH_SHORT).show();
        }
      };
      countriesList.setOnItemClickListener(itemListener);
    }
}
```

Контрольные вопросы:

1. Как используется списки (menus) ?
2. Как используется виды(views) ?
3. Как используется рисунки (images) ?
4. Опишите галерею и ImageView.
5. Опишите ImageSwitcher.
6. Опишите GridView.