**Внешний вид и selector’ы**

Стандартные контролы вызывают у пользователей скуку, поэтому, практически все приложения в android market’e, обладают своими дизайнерскими решениями. Корпорация Google создала для этого мощный описательный инструмент стилей и элементов, основанный на xml. С его помощью можно задавать общий стиль групп элементов и рисовать свои новые контролы.

Для начала создадим в среде Eclipse новое приложение и назовем его Tic Tac Toe, установим в качестве **Build Target** — Android 1.6 и **Create activity**: TicTacToe.

Автоматически будет создан файл TicTacToe.java. Это activity, она будет вызываться при запуске приложения. Среда Android узнает об этом из файла AndroidManifest.xml, где для нашей activity задан <intent-filter>:

<activity android:name=".activities.TicTacToe"

android:label="@string/app\_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

Перейдем непосредственно к созданию интерфейса. В качестве корневого элемента main.xml мы положим ScrollView, этот элемент позволит нам прокручивать нашу activity, если контролы в горизонтальном положении телефона выйдут за пределы формы. Следующим элементом идет LinearLayout, благодаря ему мы разместим наши кнопки стройно в ряд, задав все необходимые отступы.

Теперь сделаем <selector>, для того чтобы задавать правила отображения наших кнопок. Этот xml файл будет содержать данные о том, как рисовать контрол в нажатом, сфокусированном или обычном положении.

button\_selector.xml

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

<selector xmlns:android="http:**//**schemas.android.com/apk/res/android">

<item android:state\_pressed="true">

<shape>

<solid

android:color="#774401" />

<stroke

android:width="1dp"

android:color="#171717" />

<corners

android:radius="4dp" />

<padding

android:left="10dip"

android:top="10dip"

android:right="10dip"

android:bottom="10dip" />

</shape>

</item>

<item>

<shape android:layout\_marginBottom="25dip">

<gradient

android:startColor="#9E5A02"

android:endColor="#2A1800"

android:angle="270" />

<stroke

android:width="1dp"

android:color="#171717" />

<corners

android:radius="6dp" />

<padding

android:left="10dip"

android:top="10dip"

android:right="10dip"

android:bottom="10dip" />

</shape>

</item>

</selector>

Наш selector содержит описание нажатого состояния и состояния покоя. Во втором случае мы заливаем кнопку градиентом, задавая startColor и endColor. Сorners указывает на углы нашей кнопки. Для эффекта нажатого состояния мы уменьшаем угол скругления и заменяем градиент монотонным цветом. Все имена атрибутов легко читаемы, их полный список можно посмотреть на официальном сайте [Shape Drawable](http://developer.android.com/guide/topics/resources/drawable-resource.html" \l "Shape).

Чтобы не описывать, каким будет размер шрифта и стиль текста в каждой кнопке, мы зададим один общий для всех, создав в папке values файл styles.xml.

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

<resources>

<style name="ButtonStyle">

<item name="android:layout\_width">fill\_parent</item>

<item name="android:layout\_height">wrap\_content</item>

<item name="android:textColor">#ffffff</item>

<item name="android:gravity">center</item>

<item name="android:layout\_margin">4dip</item>

<item name="android:minHeight">48dip</item>

<item name="android:textSize">18dip</item>

<item name="android:textStyle">bold</item>

<item name="android:shadowColor">#000000</item>

<item name="android:shadowDx">1</item>

<item name="android:shadowDy">1</item>

<item name="android:shadowRadius">2</item>

</style>

</resources>

Тогда xml текст кнопки будет простой:

<Button

android:id="@+id/new\_game\_button"

android:text="@string/new\_game\_label"

android:layout\_width="fill\_parent"

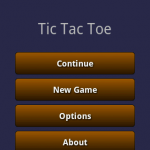
android:layout\_height="wrap\_content"

android:background="@drawable/button\_selector"

style="@style/ButtonStyle"

/>

В итоге у нас получается интересная первая activity:

[](http://www.4htc.ru/wp-content/uploads/2011/09/0_5994b_b9e677b3_L.png)

**Стратегия игры**

Классические «крестики-нолики» на поле 3×3 имеют достаточно короткое описание всех вариантов состояний. При совершенной игре результат всегда будет сведен к ничьей, если один из соперников не совершит ошибку. Для игры за компьютер (нолики) мы будем использовать простой принцип приоритета клеток. Максимальным приоритетом будет обладать центральное поле. Вторыми по очереди будут поля в углах, и самые *нежеланные*будут оставшиеся поля в середине.

Весь алгоритм можно описать так:

* Если можно сделать шаг ведущий к выигрышу — делаем. Иначе,
* Если нужно блокировать возможность выигрыша соперника — блокируем. Иначе
* Делаем ход в ячейку с максимальным приоритетом.

Три уровня сложности будут такими, для **Hard**придерживаемся вышеописанного алгоритма, к слову, он упрощен и обыграть его имеются возможности. Для **Easy**будем ставить нолик в случайную ячейку, а для **Medium**мы будем делать в такой пропорции: один из трех будет глупый ход, два из трех — умные.

Tic-tac-toe layout

Создадим game.xml layout для отображения процесса игры. Мы будем использовать TableLayout с тремя строками и столбцами для поля 3×3. В каждой ячейке будет лежать кнопка с пустой или заполненной картинкой.

<TableLayout android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" xmlns:android="http:**//**schemas.android.com/apk/res/android"

android:id="@+id/buttons" android:background="@color/background">

<TableRow

android:id="@+id/row1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="1dip"

android:layout\_marginRight="1dip"

android:layout\_marginBottom="2dip">

<Button

android:text=""

android:id="@+id/button\_11"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@drawable/clear"

android:layout\_marginRight="2dip"

android:layout\_marginLeft="12dip">

...

<Button

android:text=""

android:id="@+id/button\_33"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@drawable/clear">

</Button>

</TableRow>

…

TableRow обеспечивает горизонтальное распределение всех принадлежащих ему контролов. Чтобы наша сетка не сливалось в единое целое,  зададим необходимые отступы для каждой ячейке layout\_margin Right, Left, Top или Bottom. При нажатии на пустую кнопку, мы будет рисовать в ней «крестик». Ход компьютера будет рисовать ответный «нолик».

Model-view-controller

Теперь немного об архитектуре нашей игры. Для реализации режима битвы и возможности продолжить играть после закрытия activity, нам нужно будет хранить текущее состояние (счет, заполненные ячейки). В этом нам помогут такие шаблоны проектирования как model-view-controller для разделения логики и singleton для хранения состояния. Шаблон MVC позволит разделить данные, представление и обработку действий пользователя на три отдельных компонента, что облегчит реализацию алгоритма игры и дальнейшее сопровождение программы.

В модели (см. TicTacToeModel.java) мы реализуем два метода, doSmartMove() для хорошего хода и doStupidMove() для случайного. В случае, когда уровень сложности выбран Medium, мы будем делать два умных и один глупый ходы. Модель также будет хранить счет выигранных и проигранных боев.

Задача controller’а обновлять ход игры по запросу, а в роли view выступает сама Activity.

**Локализация**

География Android устройств обширна, поэтому вопрос локализации занимает ключевое место при разработке приложений под эту платформу. К счастью, нам не прийдется изобретать велосипед, Google позаботился о разработчиках в этом вопросе.

Правила локализации приложений таковы — мы создаем несколько наборов ресурсов, первый — по умолчанию res/values/strings.xml, остальные для нужной нам локали res/values-<qualifiers>/strings.xml, например, values-en для английского или values-ja для японского. При запуске activity Android выбирает какие ресурсы загрузить, основываясь на локали самого устройства. К слову, под ресурсами в Android подразумеваются не только текстовые строки, а также layout’ы, звуковые файлы, графика и другие статические данные.

Для нашего приложения мы создадим папки res/values, res/values-en и res/values-ru, в них будут лежать файлы strings.xml. Эти файлы содержат название приложения, сообщения и вопросы. Английский файл выглядит так:

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

<resources>

<string name="app\_name">Tic Tac Toe</string>

<string name="main\_title">Tic Tac Toe</string>

<string name="continue\_label">Continue</string>

<string name="new\_game\_label">New Game</string>

<string name="options\_label">Options</string>

<string name="easy\_label">Easy</string>

<string name="medium\_label">Medium</string>

<string name="hard\_label">Hard</string>

<string name="message\_title">Message</string>

<string name="question\_title">Question</string>

<string name="draw\_game">Draw!</string>

<string name="nought\_win\_game">Nought win!</string>

<string name="cross\_win\_game">Cross win!</string>

<string name="restart\_game">Restart Game?</string>

<string name="about\_label">About</string>

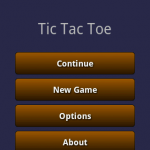
<string name="about\_title">About Tic Tac Toe</string>

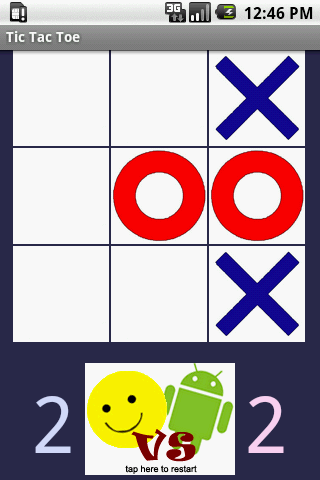
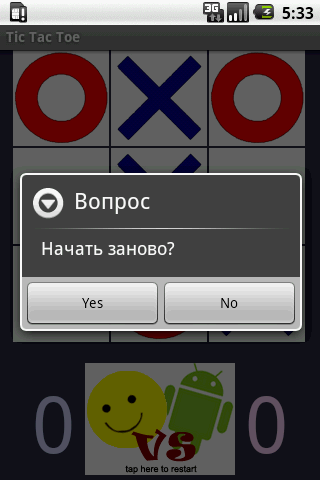
...

</resources>

Еще нюанс, android developer’s guide предупреждает нас о необходимости иметь точную копию какой-нибудь локализации в качестве ресурса по умолчанию, чаще всего — английскую. И приводит пример, что если не будет хватать строки в файле res/values/strings.xml, которая есть в res/values-en/strings.xml и используется в приложении, то возможно все будет компилироваться без проблем, но в локали отличной от английской пользователь увидит сообщение об ошибке и кнопку о закрытии приложения.

В итоге у нас получились вот такие локализованные крестики-нолики:

[](http://www.4htc.ru/wp-content/uploads/2011/09/0_5994b_b9e677b3_L.png)

[](http://www.4htc.ru/wp-content/uploads/2011/09/0_5994c_2f000f8a_L.png)[](http://www.4htc.ru/wp-content/uploads/2011/09/0_5994d_e4e63bbf_L.png)[](http://www.4htc.ru/wp-content/uploads/2011/09/0_59a03_7b395661_L.png)