**Игры с клубком. Добавляем кнопки**

Рассмотрим пример, как добавить кнопки на игровое поле.

Так как мы используем не разметку, а отдельный компонент **View**, то стандартную кнопку мы не сможем добавить на экран (на самом деле можно, но это выходит за рамки статьи). Но мы можем нарисовать свою кнопку и она будет вести себя как обычная кнопка. В принципе кнопку можно было добавить на экран с клубком, но мы поступим по-другому. Мы создадим ещё одну активность, которая тоже использовать **View** в качестве экрана и сделаем активность стартовой. А при нажатии на кнопку попадём уже на экран с клубком. Так мы создадим подобие игры, когда пользователь сначала видит начальный экран с различными кнопками и запускает либо игру, либо справку и т.д.

Нарисуем кнопку (можно скачать готовые изображения кнопок из интернета) и поместим её в папку **res/drawable**.

Сделаем копию класса **Board** и назовём её **StartScreen**. Код практически идентичен, нам нужно загрузить изображение кнопки из ресурсов и показать его на экране.

package ru.alexanderklimov.ball;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.view.View;

public class StartScreen extends View {

private Bitmap mButtonUp;

public StartScreen(Context context) {

super(context);

// TODO Auto-generated constructor stub

mButtonUp = BitmapFactory.decodeResource(getResources(),

R.drawable.button\_up);

}

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

canvas.drawBitmap(mButtonUp, (canvas.getWidth() - mButtonUp.getWidth()) / 2,

(int) (canvas.getHeight() \* 0.7), null);

}

}

Обратите внимание, что мы центрируем кнопку по горизонтали, а по вертикали выводим на высоте 70% от общей высоты холста. Такой подход удобнее, чем точно задавать координаты в пикселях, так как на разных разрешениях кнопка может оказаться не там, где изначально планировалось.

Создадим новую активность **StartActivity**, которая станет стартовой.

package ru.alexanderklimov.ball;

import android.app.Activity;

import android.os.Bundle;

public class StartActivity extends Activity {

StartScreen startScreen;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

getActionBar().hide(); // скрыть ActionBar

startScreen = new StartScreen(this);

setContentView(startScreen);

}

}

В манифесте добавим созданную активность и сделаем её главной, чтобы она запускалась первой.

<activity

android:name="ru.alexanderklimov.ball.MainActivity"

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

</activity>

<activity

android:name="ru.alexanderklimov.ball.StartActivity"

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

<intent-filter>

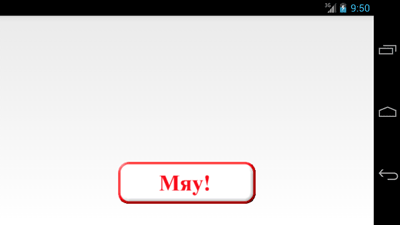
<action android:name="android.intent.action.MAIN" />

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

</intent-filter>

</activity>

Запускаем проект и видим кнопку в нужном месте.



Но если мы будем нажимать на неё, то не увидим никакой интерактивности. Поэтому нарисуем вторую кнопку для нажатого состояния. Добавим её в ресурсы и добавим код в класс**StartScreen**, совпадающий с кодом для первой кнопки.

// в классе

private Bitmap mButtonDown;

// в конструкторе

mButtonDown = BitmapFactory.decodeResource(getResources(),

R.drawable.button\_down);

В методе **onDraw()** нет смысла выводить эту кнопку на экран, она должна рисоваться при нажатии, а при отпускания пальца с экран, нужно вернуть прежнее состояние кнопки.

Добавим булевую переменную, которая будет отслеживать состояние кнопки.

private boolean mIsButtonPressed;

В первом уроке мы кратко использовали метод **onTouchEvent()** для отслеживания касаний. Снова воспользуемся этим кодом и будем отслеживать нажатие кнопки.

public boolean onTouchEvent(MotionEvent event) {

int eventaction = event.getAction();

int x = (int) event.getX();

int y = (int) event.getY();

switch (eventaction) {

case MotionEvent.ACTION\_DOWN:

if (x > (mScreenW - mButtonUp.getWidth()) / 2

&& x < ((mScreenW - mButtonUp.getWidth()) / 2)

+ mButtonUp.getWidth()

&& y > (int) (mScreenH \* 0.7)

&& y < (int) (mScreenH \* 0.7) + mButtonUp.getHeight()) {

mIsButtonPressed = true;

}

break;

case MotionEvent.ACTION\_MOVE:

break;

case MotionEvent.ACTION\_UP:

mIsButtonPressed = false;

break;

}

invalidate();

return true;

}

Получилось очень длинное условие, которое проверяет, что касание произошло в области кнопки. Если условие выполняется, то переменной **mIsButtonPressed** присваивается значение**true**. В момент отрыва пальца от экрана, присваиваем этой переменной значение **false**.

Теперь в методе **onDraw()** мы можем выводить разные изображения кнопки в зависимости от значения булевой переменной.

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

// canvas.drawBitmap(mButtonUp,

// (canvas.getWidth() - mButtonUp.getWidth()) / 2,

// (int) (canvas.getHeight() \* 0.7), null);

if (mIsButtonPressed) {

canvas.drawBitmap(mButtonDown,

(mScreenW - mButtonUp.getWidth()) / 2,

(int) (mScreenH \* 0.7), null);

} else {

canvas.drawBitmap(mButtonUp, (mScreenW - mButtonUp.getWidth()) / 2,

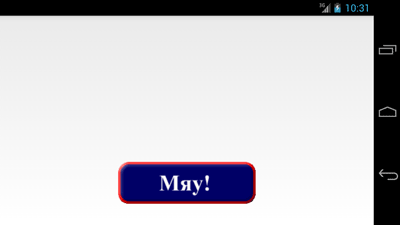
(int) (mScreenH \* 0.7), null);

}

}

На этот раз я вычисляю размеры экрана не через методы холста (закоментировано), а через отдельные переменные **mScreenW** и **mScreenH**, значения которых вычисляютс в методе**onSizeChanged()** (см. исходники).

Запустим проект, чтобы убедиться, что кнопка меняет своё состояние при нажатии.



Вроде работает, но ничего не происходит. А нам нужно переключиться на экран с клубком. Это можно сделать через намерение. Добавляем переменную для контекста и подключаем её в конструкторе.

private Context mContext;

public StartScreen(Context context) {

super(context);

mContext = context;

// предыдущий код

}

Получив контекст, мы можем запустить намерение. Сделаем это в момент отпускания пальца.

case MotionEvent.ACTION\_UP:

if (mIsButtonPressed) {

Intent intent = new Intent(mContext, MainActivity.class);

mContext.startActivity(intent);

}

mIsButtonPressed = false;

break;

}

Теперь при нажатии на кнопку, она не только меняет своё состояние, но и выполняет действие - переключается на активность с клубком.

Хочу ещё раз подчеркнуть, что для стартового экрана вы могли воспользоваться стандартным способом с использованием XML-макета с нужным дизайном. В этом примере был показан принцип добавления кнопок на экран класса **View**. Например, можно добавить такую кнопку на экран с клубком, чтобы остановить движение клубка по экрану.

Исходник класса

package ru.alexanderklimov.ball;

import android.content.Context;

import android.content.Intent;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.view.MotionEvent;

import android.view.View;

public class StartScreen extends View {

private Bitmap mButtonUp;

private Bitmap mButtonDown;

private boolean mIsButtonPressed;

private int mScreenW;

private int mScreenH;

private Context mContext;

public StartScreen(Context context) {

super(context);

// TODO Auto-generated constructor stub

mContext = context;

mButtonUp = BitmapFactory.decodeResource(getResources(),

R.drawable.button\_up);

mButtonDown = BitmapFactory.decodeResource(getResources(),

R.drawable.button\_down);

}

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

// canvas.drawBitmap(mButtonUp,

// (canvas.getWidth() - mButtonUp.getWidth()) / 2,

// (int) (canvas.getHeight() \* 0.7), null);

if (mIsButtonPressed) {

canvas.drawBitmap(mButtonDown,

(mScreenW - mButtonUp.getWidth()) / 2,

(int) (mScreenH \* 0.7), null);

} else {

canvas.drawBitmap(mButtonUp, (mScreenW - mButtonUp.getWidth()) / 2,

(int) (mScreenH \* 0.7), null);

}

}

public boolean onTouchEvent(MotionEvent event) {

int eventaction = event.getAction();

int x = (int) event.getX();

int y = (int) event.getY();

switch (eventaction) {

case MotionEvent.ACTION\_DOWN:

if (x > (mScreenW - mButtonUp.getWidth()) / 2

&& x < ((mScreenW - mButtonUp.getWidth()) / 2)

+ mButtonUp.getWidth()

&& y > (int) (mScreenH \* 0.7)

&& y < (int) (mScreenH \* 0.7) + mButtonUp.getHeight()) {

mIsButtonPressed = true;

}

break;

case MotionEvent.ACTION\_MOVE:

break;

case MotionEvent.ACTION\_UP:

if (mIsButtonPressed) {

Intent intent = new Intent(mContext, MainActivity.class);

mContext.startActivity(intent);

}

mIsButtonPressed = false;

break;

}

invalidate();

return true;

}

@Override

public void onSizeChanged(int w, int h, int oldw, int oldh) {

super.onSizeChanged(w, h, oldw, oldh);

mScreenW = w;

mScreenH = h;

}

}