**Движение картинки**

Неподвижный клубок в центре экрана из прошлого урока не слишком интересен. Надо заставить клубок двигаться по экрану.

Добавим в класс **Board** две новые переменные для установки координат клубка и в конструкторе присвоим им нулевые значения.

private int mPosX, mPosY;

// В конструкторе

mPosX = mPosY = 0;

Давайте пока будем двигать клубок по горизонтали и задействуем только первую переменную **mPosX**. Чтобы сдвинуть клубок на 10 единиц вправо, надо к текущей позиции прибавить значение 10. Так и сделаем. Внесём изменения в метод **onDraw()**:

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPosX += 10;

canvas.drawBitmap(mBallBitmap, mPosX, canvas.getHeight() / 2 - mBallBitmap.getHeight()/ 2, null);

Log.i("Где клубок?", String.valueOf(mPosX));

}

Что произойдёт при запуске? К значению 0 добавится значение 10 и клубок сдвинется чуть вправо. Скорее всего вы даже не заметите движение. Заметьте, что в данном случае мы не вычисляем середину ширины экрана, так как мы хотим динамически менять позицию. Конструкция **Log.i()** поможет нам получать полезную информацию в следующих опытах, поэтому мы её заранее приготовили.

А как заставить клубок сдвинуться ещё на 10 единиц и продолжать движение без остановки. Здесь следует сделать небольшое отступление и немного пояснить принцип работы метода **onDraw()**. Попробуйте представить работу метода как срабатывание затвора фотоаппарата для создания снимка. Вы сделали первый снимок движущего объекта. Что нужно сделать, чтобы запечатлеть объект на новом месте? Правильно, сделать второй снимок. А ещё лучше - сделать серию снимков. И если эти снимки быстро просматривать, то возникнет иллюзия движения. В качестве затвора служит специальный метод **invalidate()**, который накладывает новые снимки поверх старых, вызывая снова метод **onDraw()**. Проверим.

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPosX += 10;

canvas.drawBitmap(mBallBitmap, mPosX, canvas.getHeight() / 2 – mBallBitmap.getHeight()

/ 2, null);

**invalidate();**

Log.i("Где клубок?", String.valueOf(mPosX));

}

Отлично, клубок покатился. Затвор сработал, метод **onDraw()** был вызван, к значению **mPosX** прибавилось ещё 10 единиц и так далее.

Но тут возникла проблема. Клубок укатился за край экрана, но не остановился. Вот тут нам и пригодился **Log.i()**. Мы видим, что значения **mPosX** продолжают увеличиваться и клубок катится где-то в параллельной вселенной. Вероятно, когда значение станет слишком большим (попробуйте предположить, когда именно), программа сломается. Нас это не устраивает.

А что если поставить условие, что когда клубок достигнет края экрана, то значение текущей позиции клубка обнулить и всё начать сначала? Пробуем.

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

if (mPosX < canvas.getWidth()) {

mPosX += 10;

} else {

mPosX = 0;

}

canvas.drawBitmap(mBallBitmap, mPosX, canvas.getHeight() / 2 - mBallBitmap.getHeight() / 2, null);

invalidate();

Log.i("Где клубок?", String.valueOf(mPosX));

}

Мяу, клубок катится вправо, а потом снова появляется слева и повторяет движение. И так до бесконечности. Шайтан-машина, однако.

Наверное, вы и сами догадаетесь, что если зафиксировать значение переменной **mPosX** и проделать те же операции с переменной **mPosY**, то клубок будет двигаться сверху вниз. Попробуйте проверить самостоятельно.

Пример получился красивый, но не совсем реалистичный. В реальном мире клубок, достигнув края площадки, должен покатиться назад. Давайте подумаем, как это сделать.

Во-первых, заменим значение 10 переменной и назовём её **mVelocity** (скорость). Это общепринятое название для обозначения скорости перемещения объекта по экрану, хотя вы можете придумать и своё имя. Обратите внимание, что одновременно эта переменная указывает на направление. Если значение увеличивается, то объект сдвигается вправо, если уменьшается - влево. Второй момент - мы знаем, что точкой отсчёта объекта является левая верхняя точка. Но объект также имеет ширину (и высоту, естественно). Когда мы двигаем объект вправо, нам надо ждать удара не в тот момент, когда крайняя левая точка достигнет стенки (по сути, клубок выкатывается за край экрана), а когда крайняя правая точка объекта достигнет стенки. То есть, мы должны к точке отсчёта прибавить ширину самого объекта. При ударе об левую стенку экрана такой проблемы не возникает. Получится следующий код:

package ru.alexanderklimov.ball;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.util.Log;

import android.view.View;

public class Board extends View {

private Bitmap mBallBitmap;

private int mPosX, mPosY; // координаты клубка

private int mVelocity = 10; // скорость

private int mBallWidth; // ширина клубка

public Board(Context context) {

super(context);

mBallBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.klubok);

mPosX = mPosY = 0;

// Получим ширину клубка

mBallWidth = mBallBitmap.getWidth();

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPosX += mVelocity;

if (mPosX < 0) {

mVelocity = -mVelocity; // меняем на противоположное

mPosX = 0; // устанавливаем позицию при достижении стенки

} else if (mPosX + mBallWidth > canvas.getWidth()) {

mVelocity = -mVelocity;

mPosX = canvas.getWidth() - mBallWidth;

}

canvas.drawBitmap(mBallBitmap, mPosX, canvas.getHeight() / 2 – mBallBitmap.getHeight() / 2, null);

invalidate();

Log.i("Где клубок?", String.valueOf(mPosX));

}

}

Запустим проект и видим, как клубок катается туда-сюда.

Движение в одну сторону

Скрытие заголовка

ActionBar bar = getSupportActionBar();  
bar.hide();