**Отражаемся от всех стенок**

Общий принцип движения клубка из прошлого урока, надеюсь, вам стал понятен. Но прямолинейное движение по горизонтали или вертикали тоже не слишком интересно для нас. А как заставить двигаться объект по диагонали, отражаясь от всех стенок. Очевидно, что мы должны теперь задействовать две переменные, отвечающие за координаты объекта.

Пока упростим задачу и будем двигать объект сверху вниз по диагонали.

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

if (mPosX < canvas.getWidth()) {

mPosX += mVelocity;

} else {

mPosX = 0;

}

if (mPosY < canvas.getHeight()) {

mPosY += mVelocity;

}else {

mPosY = 0;

}

canvas.drawBitmap(mBallBitmap, mPosX, mPosY, null);

invalidate();

}

Это напоминает движение капель косого дождя. Клубок появляется сверху, но из разных позиций. Угол наклона зависит от нашей переменной **velocity**. Если скорость движения по горизонтали и вертикали совпадает, то падение происходит под углом 45 градусов.

Осталось рассмотреть последний пример, когда клубок отражается от всех стенок игровой площадки. Принцип точно такой же, как отражение от боковых вертикальных стенок, только добавляются дополнительные условия.

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 Board extends View {

private Bitmap mBallBitmap;

private int mPosX, mPosY; // координаты клубка

private int mVelocityX = 10; // скорость

private int mVelocityY = 10;

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 += mVelocityX;

mPosY += mVelocityY;

// Если коснулись левой стенки, то меняем направление

if (mPosX < 0) {

mPosX = 0;

mVelocityX = -mVelocityX;

}

// Если коснулись правой стенки

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

mPosX = canvas.getWidth() - mBallWidth;

mVelocityX = -mVelocityX;

}

// Если коснулись верха, то меняем направление

if (mPosY < 0) {

mPosY = 0;

mVelocityY = -mVelocityY;

}

// Если коснулись низа

if (mPosY + mBallWidth > canvas.getHeight()) {

mPosY = canvas.getHeight() - mBallWidth;

mVelocityY = -mVelocityY;

}

canvas.drawBitmap(mBallBitmap, mPosX, mPosY, null);

invalidate();

}

}

Теперь одной переменной **mVelocity** нам будет недостаточно. Клубок, коснувшись правой стенки, может отразиться как вверх, так и вниз. Зависит от движения клубка в момент удара. При этом значения **mPosX** начинают уменьшаться, но значения **mPosY** по-прежнему увеличиваются. Чтобы управлять движением в двух направлениях, приходится создавать две переменные**mVelocityX** и **mVelocityY**.

Теперь всё работает, как надо. Даже мой кот, который сидел у монитора во время написания статьи, с интересом наблюдал за движением клубка на эмуляторе. Значит, всё правильно сделали.

Вы можете самостоятельно нарисовать ещё одну стенку в центре экрана, чтобы клубок отражался не только от боковых стенок, но и от препятствия в центре. Для этого вам надо запомнить координаты препятствия и создать новые условия в коде.

**import** android.content.Context;  
**import** android.graphics.Bitmap;  
**import** android.graphics.BitmapFactory;  
**import** android.graphics.Canvas;  
**import** android.graphics.Color;  
**import** android.graphics.Matrix;  
**import** android.view.View;  
  
**public class** Board **extends** View {  
  
 **private** Bitmap **mBallBitmap**;  
  
 **private int mPosX**, **mPosY**; *// координаты клубка* **private int mVelocityX** = 10; *// скорость* **private int mVelocityY** = 10;  
 **private int mBallWidth**; *// ширина клубка* **private** Matrix **m**;  
 **float alfa**=0, **dA**=2;  
  
  
 **public** Board(Context context) {  
 **super**(context);  
 **mBallBitmap** = BitmapFactory.*decodeResource*(getResources(), R.drawable.***boll2***);  
  
 **mPosX** = **mPosY** = 0;  
 **m** = **new** Matrix();  
 *// Получим ширину клубка* **mBallWidth** = **mBallBitmap**.getWidth();  
 }  
  
 @Override  
 **protected void** onDraw(Canvas canvas) {  
 **super**.onDraw(canvas);  
 canvas.drawColor(Color.***GREEN***);  
 **mPosX** += **mVelocityX**;  
 **mPosY** += **mVelocityY**;  
  
 **alfa** +=**dA**; *//угол поворота объекта* **if**(**alfa**==360) **alfa** = 0;  
  
 *// Если коснулись левой стенки, то меняем направление* **if** (**mPosX** < 0) {  
 **mPosX** = 0;  
 **mVelocityX** = -**mVelocityX**;  
 **dA** = -2;  
 }  
  
 *// Если коснулись правой стенки* **if** (**mPosX** + **mBallWidth** > canvas.getWidth()) {  
 **mPosX** = canvas.getWidth() - **mBallWidth**;  
 **mVelocityX** = -**mVelocityX**;  
 **dA** = 2;  
 }  
  
 *// Если коснулись верха, то меняем направление* **if** (**mPosY** < 0) {  
 **mPosY** = 0;  
 **mVelocityY** = -**mVelocityY**;  
 **dA**=-2;  
 }  
  
 *// Если коснулись низа* **if** (**mPosY** + **mBallWidth** > canvas.getHeight()) {  
 **mPosY** = canvas.getHeight() - **mBallWidth**;  
 **mVelocityY** = -**mVelocityY**;  
 **dA**=2;  
 }  
**m**.setRotate(**alfa**, **mBallBitmap**.getWidth()/2, **mBallBitmap**.getHeight()/2);  
 **m**.postTranslate(**mPosX**, **mPosY**);  
 canvas.drawBitmap(**mBallBitmap**, **m**, **null**);  
invalidate();  
 }  
}

