Работаем с графикой. Основы

Цель нашего урока - понять основы графики. Мы напишем простую рисовалку - хотя это слишком громко сказано. Пока мы сами рисовать ничего не будем - за нас это будет делать глупая машина, т.е. Android. Но тем не менее некоторые полезные вещи мы узнаем, а значит повысим свой профессиональный уровень. Продолжить своё обучение можно в разделе [Котошоп](http://developer.alexanderklimov.ru/android/catshop/).

Создадим новый проект **SimplePaint**. Далее в нём создадим новый класс **Draw2D**, который будет наследоваться от **View**. Именно в этом классе мы и будем проводить графические опыты. Щёлкаем правой кнопкой мыши на имени пакета и выбираем в меню **New | Java Class**. В открывшемся диалоговом окне устанавливаем имя для класса **Draw2D**.

Добавляем код.

package ru.alexanderklimov.simplepaint; // у вас будет свой пакет

public class Draw2D extends View{

public Draw2D(Context context) {

super(context);

}

@Override

protected void onDraw(Canvas canvas){

super.onDraw(canvas);

}

}

В данном коде мы наследуемся от класса **android.view.View** и переопределяем метод класса **onDraw(Canvas canvas)**.

Далее необходимо загрузить созданный класс при старте программы. Открываем основной файл активности **MainActivity** и заменяем строчку после **super.onCreate(savedInstanceState)**:

// эта строчка нам не нужна

setContentView(R.layout.activity\_main);

Draw2D draw2D = new Draw2D(this);

setContentView(draw2D);

В нашем случае мы говорим системе, что не нужно загружать разметку в экран активности. Вместо неё мы загрузим свой класс, у которого есть свой холст для рисования.

Подготовительные работы закончены. Перейдём к графике. Весь дальнейший код мы будем писать в классе **Draw2D**. Совсем коротко о теории рисования. Для графики используется холст **Canvas** - некая графическая поверхность для рисования. Прежде чем что-то рисовать, нужно определить некоторые параметры - цвет, толщина, фигура. Представьте себе, что вы рисуете на бумаге и в вашем распоряжении есть цветные карандаши, фломастеры, кисть, циркуль, ластик и т.п. Например, вы берёте толстый красный фломастер и рисуете жирную линию, затем берёте циркуль с жёлтым карандашом и рисуете окружность. Улавливаете аналогию? Теория закончена.

Вся работа с графикой происходит в методе **onDraw()** класса **Draw2D**. Создадим виртуальную кисть в классе. В методе укажем, что будем закрашивать всю поверхность белым цветом:

private Paint mPaint = new Paint();

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

// стиль Заливка

mPaint.setStyle(Paint.Style.FILL);

// закрашиваем холст белым цветом

mPaint.setColor(Color.WHITE);

canvas.drawPaint(mPaint);

}

Итак, холст готов. Далее начинается собственно рисование. Следуя описанному выше принципу мы задаём перед каждым рисованием свои настройки и вызываем нужный метод. Например, для того, чтобы нарисовать жёлтый, круг мы включаем режим сглаживания, устанавливаем жёлтый цвет и вызываем метод **drawCircle()** с нужными координатами и заливаем окружность выбранным цветом. Получилось симпатичное солнышко.

// Рисуем желтый круг

mPaint.setAntiAlias(true);

mPaint.setColor(Color.YELLOW);

canvas.drawCircle(950, 30, 25, mPaint);

Всегда соблюдайте очерёдность рисования. Если вы поместите данный код до заливки холста белым цветом, то ничего не увидите. У вас получится, что вы сначала нарисовали на стене солнце, а потом заклеили рисунок обоями.

Для рисования зеленого прямоугольника мы также задаём координаты и цвет. У нас получится красивая лужайка.

// Рисуем зелёный прямоугольник

mPaint.setColor(Color.GREEN);

canvas.drawRect(20, 650, 950, 680, mPaint);

Далее выведем текст поверх лужайки, чтобы все видели, что она предназначена только для котов. Устанавливаем синий цвет, стиль заливки, режим сглаживания и размер прямоугольника, в который будет вписан наш текст.

// Рисуем текст

mPaint.setColor(Color.BLUE);

mPaint.setStyle(Paint.Style.FILL);

mPaint.setAntiAlias(true);

mPaint.setTextSize(32);

canvas.drawText("Лужайка только для котов", 30, 648, mPaint);

При желании можно вывести текст под углом. Пусть это будет лучик солнца.

// Текст под углом

int x = 810;

int y = 190;

mPaint.setColor(Color.GRAY);

mPaint.setTextSize(27);

String str2rotate = "Лучик солнца!";

// Создаем ограничивающий прямоугольник для наклонного текста

// поворачиваем холст по центру текста

canvas.rotate(-45, x + mRect.exactCenterX(), y + mRect.exactCenterY());

// Рисуем текст

mPaint.setStyle(Paint.Style.FILL);

canvas.drawText(str2rotate, x, y, mPaint);

// восстанавливаем холст

canvas.restore();

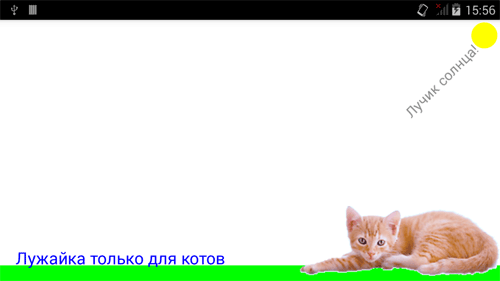
И завершим нашу композицию выводом рисунка из ресурсов.

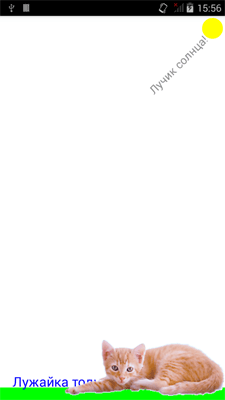
// Выводим изображение

canvas.drawBitmap(mBitmap, 450, 530, mPaint);

В данном примере я вручную подбирал размеры и координаты фигур для своего экрана. В реальных приложениях необходимо сначала вычислить размеры экрана у пользователя, а потом уже выводить фигуры в соответствии с полученными результатами. Иначе получится так, что некоторые элементы композиции просто не попадут на экран при вращении устройства. Допустим, в альбомном режиме вы установите у точки X значение 800, но в портретном режиме ширина экрана будет, скажем, 480, и точка окажется вне поле зрения. Поэтому следует позаботиться о вычислениях размеров экрана и плясать от этой печки. Ниже представлен немного переделанный вариант для общего понимания.

Финальный рисунок выглядит следующим образом в двух ориентациях. Вы можете доработать приложение, уменьшив размеры кота и т.д.





Похоже?

Исходный код класса

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

package ru.alexanderklimov.simplepaint;

import android.content.Context;

import android.content.res.Resources;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Rect;

import android.view.View;

public class Draw2D extends View {

private Paint mPaint = new Paint();

private Rect mRect = new Rect();

private Bitmap mBitmap;

public Draw2D(Context context) {

super(context);

// Выводим значок из ресурсов

Resources res = this.getResources();

mBitmap = BitmapFactory.decodeResource(res, R.drawable.cat\_bottom);

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

int width = canvas.getWidth();

int height = canvas.getHeight();

// стиль Заливка

mPaint.setStyle(Paint.Style.FILL);

// закрашиваем холст белым цветом

mPaint.setColor(Color.WHITE);

canvas.drawPaint(mPaint);

// Рисуем желтый круг

mPaint.setAntiAlias(true);

mPaint.setColor(Color.YELLOW);

// canvas.drawCircle(950, 30, 25, mPaint);

canvas.drawCircle(width - 30, 30, 25, mPaint);

// Рисуем зеленый прямоугольник

mPaint.setColor(Color.GREEN);

// canvas.drawRect(20, 650, 950, 680, mPaint);

canvas.drawRect(0, canvas.getHeight() - 30, width, height, mPaint);

// Рисуем текст

mPaint.setColor(Color.BLUE);

mPaint.setStyle(Paint.Style.FILL);

mPaint.setAntiAlias(true);

mPaint.setTextSize(32);

// canvas.drawText("Лужайка только для котов", 30, 648, mPaint);

canvas.drawText("Лужайка только для котов", 30, height - 32, mPaint);

// Текст под углом

// int x = 810;

int x = width - 170;

int y = 190;

mPaint.setColor(Color.GRAY);

mPaint.setTextSize(27);

String beam = "Лучик солнца!";

canvas.save();

// Создаем ограничивающий прямоугольник для наклонного текста

// поворачиваем холст по центру текста

canvas.rotate(-45, x + mRect.exactCenterX(), y + mRect.exactCenterY());

// Рисуем текст

mPaint.setStyle(Paint.Style.FILL);

canvas.drawText(beam, x, y, mPaint);

// восстанавливаем холст

canvas.restore();

// Выводим изображение

// canvas.drawBitmap(mBitmap, 450, 530, mPaint);

canvas.drawBitmap(mBitmap, width - mBitmap.getWidth(), height - mBitmap.getHeight() - 10, mPaint);

}

}