Класс android.graphics.Canvas

[Центрируем текст](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#centertext)

Класс **Canvas** предоставляет методы для рисования, которые отображают графические примитивы на исходном растровом изображении. При этом надо сначала подготовить кисть (класс **Paint**), который позволяет указывать, как именно графические примитивы должны отображаться на растровом изображении (цвет, обводка, стиль, сглаживание шрифта и т.д.).

Android поддерживает полупрозрачность, градиентные заливки, округленные прямоугольники и сглаживание. Из-за ограниченных ресурсов векторная графика пока что не поддерживается, вместо этого используется традиционная растровая перерисовка.

1. Методы

Ниже представлены некоторые методы класса **Canvas**, которые что-то рисуют.

* drawARGB()/drawRGB()/drawColor(). Заполняет холст сплошным цветом.
* [drawArc()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawarc). Рисует дугу между двумя углами внутри заданной прямоугольной области.
* [drawBitmap()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawbitmap). Рисует растровое изображение на холсте. Вы можете изменять внешний вид целевой картинки, указывая итоговый размер или используя матрицу для преобразования.
* drawBitmapMesh(). Рисует изображение с использованием сетки, с помощью которой можно управлять отображением итоговой картинки, перемещая точки внутри неё.
* [drawCircle()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawcircle). Рисует окружность с определенным радиусом вокруг заданной точки.
* [drawLine(s)()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawline). Рисует линию (или последовательность линий) между двумя точками.
* [drawOval()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawoval). Рисует овал на основе прямоугольной области.
* [drawPaint()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawpaint). Закрашивает весь холст с помощью заданного объекта **Paint**.
* [drawPath()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawpath). Рисует указанный контур, используется для хранения набора графических примитивов в виде единого объекта.
* drawPicture(). Рисует объект **Picture** внутри заданного прямоугольника.
* [drawPoint()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawpoint). Рисует точку в заданном месте.
* drawPosText(). Рисует текстовую строку, учитывая смещение для каждого символа.
* [drawRect()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawrect). Рисует прямоугольник.
* [drawRoundRect()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawroundrect). Рисует прямоугольник с закругленными углами.
* [drawText()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawtext). Рисует текстовую строку на холсте. Шрифт, размер, цвет и свойства отображения текста задаются в соответствующем объекте **Paint**.
* drawTextOnPath(). Рисует текст, который отображается вокруг определенного контура.
* drawVertices(). Рисует набор треугольников в виде совокупности вершинных (вертексных) точек.
* [rotate() и restore()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#rotate). Вращение холста
* [Методы scale() и translate()](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#scale). Изменение и перемещение координатной системы

Мы уже изучали основы рисования в первом месяце обучения ([Работаем с графикой. Основы](http://developer.alexanderklimov.ru/android/simplepaint.php)). Можно вернуться к этому проекту, закомментировать код вывода графики и продолжить изучение рисования при помощи методов класса **Canvas**.

Метод drawArc()

Метод **drawArc()** позволяет рисовать дуги и сектора. Ниже приводится код для трёх вариантов: сектор с заливкой (похож на PacMan), сектор без заливки (контур) и часть дуги:

float width = 400f;

float height = 240f;

float radius = 100f;

Path path = new Path();

path.addCircle(width, height, radius, Path.Direction.CW);

Paint paint = new Paint();

paint.setColor(Color.WHITE); // установим белый цвет

paint.setStrokeWidth(5);

paint.setStyle(Paint.Style.FILL); // заливаем

paint.setAntiAlias(true);

float center\_x, center\_y;

center\_x = 240;

center\_y = 220;

final RectF oval = new RectF();

oval.set(center\_x - radius, center\_y - radius, center\_x + radius,

center\_y + radius);

canvas.drawArc(oval, 45, 270, true, paint); // рисуем пакмана

// рисуем большого пакмана без заливки

paint.setStyle(Paint.Style.STROKE);

oval.set(center\_x - 200f, center\_y - 200f, center\_x + 200f,

center\_y + 200f);

canvas.drawArc(oval, 45, 270, true, paint);

paint.setStyle(Paint.Style.STROKE);

// рисуем разорванное кольцо

center\_y = 540;

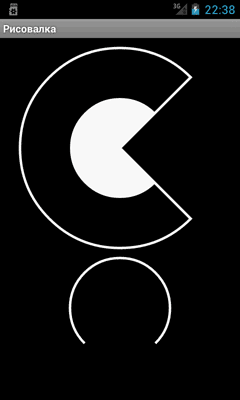
oval.set(center\_x - radius,

center\_y - radius,

center\_x + radius,

center\_y + radius);

canvas.drawArc(oval, 135, 270, false, paint);



Метод drawBitmap()

Вывести готовое изображение просто.

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

Метод drawCircle()

Первые два аргумента определяют координаты центра окружности, следующий аргумент — её радиус в пикселах, последний — объект **Paint**.

Нарисуем зелёный круг.

float w, h, cx, cy, radius;

w = getWidth();

h = getHeight();

cx = w / 2;

cy = h / 2;

if (w > h) {

radius = h / 4;

} else {

radius = w / 4;

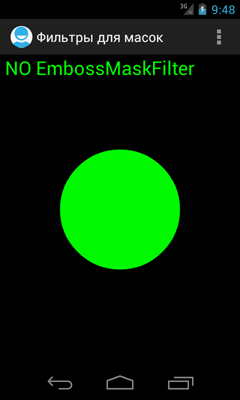
}

Paint paint = new Paint();

paint.setColor(Color.GREEN); // установим зеленый цвет

paint.setStyle(Paint.Style.FILL);

canvas.drawCircle(cx, cy, radius, paint);



drawLine(s)()

Простой метод - указываем начальные и конечные координаты отрезка.

package ru.alexanderklimov.simplepaint;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.view.View;

public class Draw2D extends View {

private Paint mPaint;

public Draw2D(Context context) {

super(context);

init();

}

private void init() {

mPaint = new Paint();

mPaint.setColor(Color.BLUE);

mPaint.setStrokeWidth(10);

}

@Override

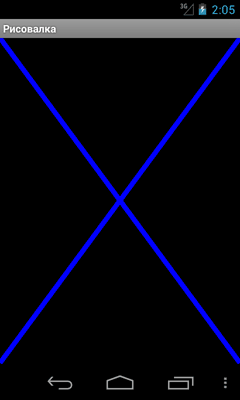
protected void onDraw(Canvas canvas) {

canvas.drawLine(0, 0, 480, 650, mPaint);

canvas.drawLine(480, 0, 0, 650, mPaint);

}

}



Метод drawOval()

Метод **drawOval()** рисует овалы. Естественно, если вы зададите одинаковые размеры ширины и высоты, то получите круг/окружность.

@Override

protected void onDraw(Canvas canvas) {

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setColor(Color.GRAY);

mPaint.setAntiAlias(true);

// овал по всему экрану

RectF oval1 = new RectF(0, 0, getWidth(), getHeight());

canvas.drawOval(oval1, mPaint);

// овал в виде круга (одинаковые размеры высоты и ширины)

mPaint.setStyle(Paint.Style.FILL);

mPaint.setColor(Color.RED);

RectF oval2 = new RectF(50, 50, 150, 150);

canvas.drawOval(oval2, mPaint);

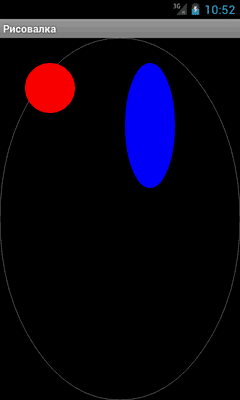
// маленький синий овал

mPaint.setColor(Color.BLUE);

RectF oval3 = new RectF(250, 50, 350, 300);

canvas.drawOval(oval3, mPaint);

}



Если вам нужно наклонить овал в ту или иную сторону, то поверните холст на требуемый угол с помощью метода **rotate()**. Не забудьте потом повернуть холст обратно, что следующие фигуры выводились нормально.

Повернём синий овал из предыдущего примера:

float rotate\_center\_x = 200;

float rotate\_center\_y = 200;

float rotate\_angle = 45;

// поворачиваем холст

canvas.rotate(-rotate\_angle, rotate\_center\_x, rotate\_center\_y);

// маленький синий овал

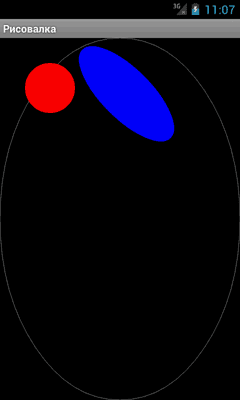
paint.setColor(Color.BLUE);

RectF oval3 = new RectF(250, 50, 350, 300);

canvas.drawOval(oval3, paint);

// возвращаем холст на прежний угол

canvas.rotate(rotate\_angle, rotate\_center\_x, rotate\_center\_y);



Метод drawPaint()

Метод позволяет закрасить весь холст одним цветом.

@Override

protected void onDraw(Canvas canvas) {

// закрашиваем холст жёлтым цветом

mPaint.setColor(Color.YELLOW);

// стиль Заливка

mPaint.setStyle(Paint.Style.FILL);

canvas.drawPaint(mPaint);

}

Метод drawRect()

У метода существует три перегруженные версии для рисования прямоугольника. Рассмотрим один из них:

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

// создаем пустой прямоугольник и задаем координаты верхней левой и нижней правой точек

Rect myRect = new Rect();

myRect.set(0, canvas.getHeight()/2, canvas.getWidth(), canvas.getHeight());

// кисть

Paint greenPaint = new Paint();

// цвет кисти - зеленый

greenPaint.setColor(Color.GREEN);

// тип - заливка

greenPaint.setStyle(Paint.Style.FILL);

// закрашиваем зелёным прямоугольником вторую половину экрана

canvas.drawRect(myRect, greenPaint);

}

Метод drawRoundRect()

Для рисования прямоугольников с закруглёнными углами используется метод **drawRoundRect (RectF rect, float rx, float ry, Paint paint)**.

В параметрах указываются ограничивающий прямоугольник, радиусы овалов для скругления углов и кисть.

Реализуем три разных способа:

float width = (float) getWidth();

float height = (float) getHeight();

float radius;

if (width > height) {

radius = height / 4;

} else {

radius = width / 4;

}

Paint paint = new Paint();

paint.setAntiAlias(true);

paint.setColor(Color.WHITE);

paint.setStrokeWidth(5);

paint.setStyle(Paint.Style.FILL);

float center\_x, center\_y;

center\_x = width / 4;

center\_y = height / 4;

final RectF rect = new RectF();

rect.set(center\_x - radius, center\_y - radius, center\_x + radius,

center\_y + radius);

canvas.drawRoundRect(rect, 50, 50, paint);

paint.setStyle(Paint.Style.STROKE);

center\_x = width / 2;

center\_y = height / 2;

rect.set(center\_x - radius,

center\_y - radius, center\_x + radius, center\_y + radius);

canvas.drawRoundRect(rect, 100, 50, paint);

paint.setStyle(Paint.Style.STROKE);

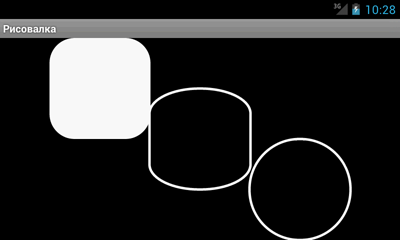
center\_x = width \* 3 / 4;

center\_y = height \* 3 / 4;

rect.set(center\_x - radius, center\_y - radius, center\_x + radius,

center\_y + radius);

canvas.drawRoundRect(rect, 100, 100, paint);



Метод drawPath()

Для рисования соединённых отрезков можно использовать метод **drawPath()**, указав в параметрах настройки для рисования и массив координат точек. Для удобства добавим в класс **Draw2D** новый класс Pt, который позволит быстро создать массив точек с заданными координатами. Далее настраиваем объекты для рисования и формируем путь через созданный массив. В результате получим кошкин дом.

// Класс для создания точки

class Pt {

float x, y;

Pt(float \_x, float \_y) {

x = \_x;

y = \_y;

}

}

// создаем массив точек

Pt[] myPath = { new Pt(300, 50), new Pt(200, 200), new Pt(200, 400),

new Pt(400, 400), new Pt(400, 200), new Pt(300, 50), };

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

Paint paint = new Paint();

paint.setColor(Color.WHITE);

paint.setStrokeWidth(3);

paint.setStyle(Paint.Style.STROKE);

Path path = new Path();

// переходим в первую точку рисования

path.moveTo(myPath[0].x, myPath[0].y);

// рисуем отрезки по заданным точкам

for (int i = 1; i < myPath.length; i++){

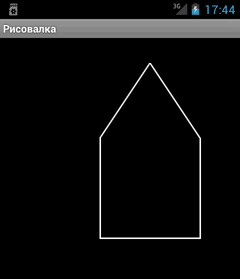
path.lineTo(myPath[i].x, myPath[i].y);

}

// выводим результат

canvas.drawPath(path, paint);

}



Путь можно составлять не только из точек, но и из фигур, например, дуг. Сначала формируем дугу, добавляем её в путь при помощи метода**Path.addArc()**, повторяем операцию снова несколько раз, а в конце выводим окончательынй вариант:

float radius = 150;

Paint paint = new Paint();

paint.setAntiAlias(true);

paint.setColor(Color.WHITE);

paint.setStrokeWidth(5);

paint.setStyle(Paint.Style.STROKE);

Path path = new Path();

float center\_x, center\_y;

center\_x = 0;

center\_y = 0;

final RectF oval = new RectF();

oval.set(center\_x - radius, center\_y - radius, center\_x + radius,

center\_y + radius);

path.addArc(oval, 10, 70); // первая дуга

oval.set(center\_x - radius - 10, center\_y - radius - 10, center\_x + radius + 20,

center\_y + radius + 20);

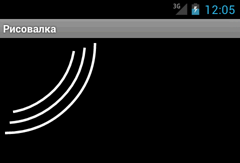
path.addArc(oval, 5, 80); // вторая дуга

oval.set(center\_x - radius- 20, center\_y - radius -20, center\_x + radius + 40,

center\_y + radius + 40);

path.addArc(oval, 0, 90); // третья дуга

canvas.drawPath(path, paint); // выводим всё вместе



Можно нарисовать символ парашюта:

Paint paint = new Paint();

paint.setAntiAlias(true);

paint.setColor(Color.WHITE);

paint.setStrokeWidth(5);

paint.setStyle(Paint.Style.STROKE);

Path path = new Path();

path.moveTo(160.0f, 240.0f);

path.lineTo(140.0f, 200.0f);

path.addArc(new RectF(140, 180, 180, 220), -180, 180);

path.lineTo(160.0f, 240.0f);

path.close();

canvas.drawPath(path, paint);



Метод drawPoint()

Простой метод для рисования точки в нужно месте указанной кистью. Для координат используются значения типа **float**.

canvas.drawPoint(1F, 0F, mPaint);

Метод drawText()

С помощью метода **drawText()** можно выводить текст в заданной позиции. Добавим сначала несколько эффектов, чтобы казалось, что текст парит над поверхностью:

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

canvas.drawColor(Color.GRAY);

Paint shadowPaint = new Paint();

shadowPaint.setAntiAlias(true);

shadowPaint.setColor(Color.WHITE);

shadowPaint.setTextSize(35.0f);

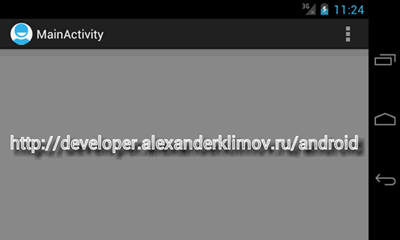
shadowPaint.setStrokeWidth(2.0f);

shadowPaint.setStyle(Paint.Style.STROKE);

shadowPaint.setShadowLayer(5.0f, 10.0f, 10.0f, Color.BLACK);

canvas.drawText("http://developer.alexanderklimov.ru/android", 20, 200, shadowPaint);

}



Центрируем текст

Есть небольшая тонкость, если вам захочется вывести текст в центре холста. Проблем с вычислением центра холста и размером текста нет. Центр можно найти, разделив пополам значения ширины и высоты холста. А ширину и высоту текста можно узнать через метод кисти **getTextBounds()**, который возвращает ограничивающий прямоугольник.

Но вычисление ширины текста через **textBounds.width();** приводит к небольшому смещению. Лучше воспользоваться методом кисти **measureText()**. Тогда текст отцентрируется точнее.

package ru.alexanderklimov.testapplication;

import android.content.Context;

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 mTextBoundRect = new Rect();

public Draw2D(Context context) {

super(context);

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

float mTextWidth, mTextHeight;

String text = "МЯЧ";

float width, height, centerX, centerY, radius;

width = getWidth();

height = getHeight();

centerX = width / 2;

centerY = height / 2;

if (width > height) {

radius = height / 4;

} else {

radius = width / 4;

}

// 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(centerX, centerY, radius, mPaint);

// Рисуем текст

mPaint.setColor(Color.BLUE);

mPaint.setStyle(Paint.Style.FILL);

mPaint.setAntiAlias(true);

mPaint.setTextSize(100);

// Подсчитаем размер текста

mPaint.getTextBounds(text, 0, text.length(), mTextBoundRect);

//mTextWidth = textBounds.width();

// Используем measureText для измерения ширины

mTextWidth = mPaint.measureText(text);

mTextHeight = mTextBoundRect.height();

canvas.drawText(text,

centerX - (mTextWidth / 2f),

centerY + (mTextHeight /2f),

mPaint

);

}

}



Методы rotate() и restore()

Холст во время рисования можно вращать. Во многих ситуациях такой приём менее затратный по ресурсам, чем рисование самого объекта под углом. Суть в следующем: вы поворачиваете холст на нужный градус, рисуете фигуру, а затем возвращаете холст на место при помощи метода **restore()**, чтобы следующие фигуры рисовались в ожидаемых местах. Иначе остальные фигуры будут рисоваться уже относительно поворота.

В примере с [овалом](http://developer.alexanderklimov.ru/android/catshop/android.graphics.canvas.php#drawoval) уже использовался данный метод. В примере [Работаем с графикой. Основы](http://developer.alexanderklimov.ru/android/simplepaint.php) мы также поворачивали холст, чтобы вывести текст под углом.

Вращение происходит вокруг начальной точки холста (0, 0). Но можно также использовать перегруженную версию метода **rotate(float degrees, float px, float py)**, в которой можно указать координаты точки поворота.

Методы scale() и translate()

Стандартная система координат начинает свой отсчёт с верхнего левого угла. Иногда, для рисования сложных фигур удобнее назначить свою систему координат. Например, для рисования циферблата часов удобнее рисовать относительно центра экрана в диапазоне от -1 до 1.

Чтобы установить свою систему координат, нужно произвести трансформацию. В следующем примере мы установим координаты в диапазоне от 0 до 10 и нарисуем график в стандартном виде из точки 0,0 в левом нижнем углу в точку 10,10 в верхнем правом углу.

@Override

protected void onDraw(Canvas canvas) {

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setColor(Color.GRAY);

mPaint.setAntiAlias(true);

canvas.save();

float xMax = 10F;

float xMin = 0F;

float yMax = 10F;

float yMin = 0F;

float width = canvas.getWidth();

float height = canvas.getHeight();

canvas.scale(width / (xMax - xMin), -height / (yMax - yMin));

canvas.translate(-xMin + 0.2F, -yMax + 0.2F);

mPaint.setStrokeWidth(.1f);

// Ось X

canvas.drawLine(0F, 0F, 10F, 0F, mPaint);

// Ось Y

canvas.drawLine(0F, 0F, 0F, 10F, mPaint);

mPaint.setColor(Color.RED);

canvas.drawLine(0F, 0F, 10F, 10F, mPaint);

// Точки вдоль оси X

canvas.drawPoint(1F, 0F, mPaint);

canvas.drawPoint(2F, 0F, mPaint);

canvas.drawPoint(3F, 0F, mPaint);

// Точки вдоль оси Y

canvas.drawPoint(0F, 1F, mPaint);

canvas.drawPoint(0F, 2F, mPaint);

canvas.drawPoint(0F, 3F, mPaint);

canvas.restore();

}

Для наглядности я добавил на оси несколько точек. Следует обратить внимание, что мы задали диапазон от 0 до 10 и все размеры должны масштабироваться в новых величинах, в том числе и ширина обводки в методе **setStrokeWidth()**. Поэтому значения должны быть достаточно маленькими, иначе толщина обводки может просто оказаться больше самой фигуры. Кстати, в некоторых случаях с текстом и другими методами рисование масштабирование может сыграть злую шутку и дробные значения не позволят увидеть текст и некоторые линии. В этих случах приходиться создавать цепочку преобразований, когда временно масштаб увеличивается до нормальных размеров, рисуется текст с подходящим размером шрифта, затем опять всё уменьшается и т.д. Это долгая история.



Реклама