**Класс android graphics Path**

[Методы](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#methods)  
[DashPathEffect](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#dashpatheffect)  
[CornerPathEffect](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#cornerpatheffect)  
[PathDashPathEffect](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#pathdashpatheffect)  
[DiscretePathEffect](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#discretepatheffect)  
[SumPathEffect](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#sumpatheffect)  
[ComposePathEffect](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#composepatheffect)

Класс **Path** (контур) позволяет создавать прямые, кривые, узоры и прочие линии. Готовый путь затем можно вывести на экран при помощи метода **canvas.drawPath(path, paint)**.

Рассмотрим базовый пример с применением некоторых методов класса.

Создадим новый класс, наследующий от View:

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Path;

import android.util.AttributeSet;

import android.view.View;

public class PathView extends View {

private Paint mPaint;

private Path mPath;

public PathView(Context context) {

super(context);

// TODO Auto-generated constructor stub

init();

}

public PathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mPaint = new Paint();

mPaint.setStyle(Paint.Style.STROKE);

mPath = new Path();

}

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

mPaint.setAntiAlias(true);

mPaint.setColor(Color.RED);

mPaint.setStrokeWidth(3);

mPath.moveTo(50, 50);

mPath.cubicTo(300, 50, 100, 400, 400, 400);

canvas.drawPath(mPath, mPaint);

mPath.reset();

mPaint.setColor(Color.GREEN);

mPaint.setStrokeWidth(1);

mPath.moveTo(50, 50);

mPath.lineTo(300, 50);

mPath.lineTo(100, 400);

mPath.lineTo(400, 400);

canvas.drawPath(mPath, mPaint);

}

}

Подключим его к главной активности, чтобы вывести на экран:

package ru.alexanderklimov.path;

import android.app.Activity;

import android.os.Bundle;

public class MainActivity extends Activity {

@Override

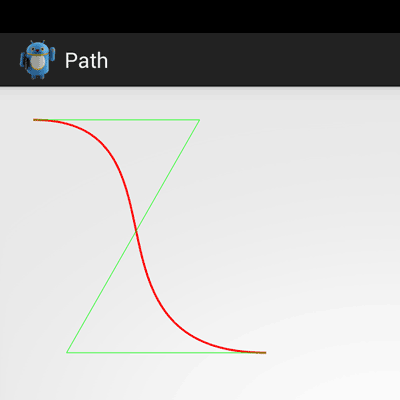
protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(new PathView(this));

}

}



**Методы**

Метод **reset()** очищает объект **Path**.

Метод **moveTo()** ставит кисть в указанную точку, с которой пойдёт новая линия.

Метод **lineTo()** рисует линию от текущей точки до указанной, следующее рисование пойдёт уже от указанной точки.

Метод **close()** закрывает контур.

Методы **addRect()**, **addCircle()** добавляю к контуру прямоугольник и окружность. В методах используется параметр, отвечающий за направление. Есть два варианта: **Path.Direction.CW** (по часовой) и **Path.Direction.CCW** (против часовой).

Метод **cubicTo()** рисует кубическую кривую Безье. По аналогии можете изучить другие методы.

Методы **moveTo()**, **lineTo()**, **quadTo()**, **cubicTo()** имеют методы-двойники, начинающиеся с буквы **r** (relative): **rMoveTo()**, **rLineTo()**, **rQuadTo()**, **rCubicTo()**. Данные методы используют не абсолютные, а относительные координаты.

Спроектируем лестницу при помощи метода **lineTo()**.

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setAntiAlias(true);

mPaint.setColor(Color.RED);

mPaint.setStrokeWidth(3);

mPath.moveTo(50, 50);

mPath.lineTo(150, 50);

mPath.lineTo(150, 100);

mPath.lineTo(250, 100);

mPath.lineTo(250, 150);

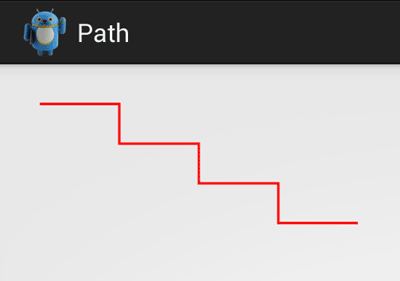
mPath.lineTo(350, 150);

mPath.lineTo(350, 200);

mPath.lineTo(450, 200);

canvas.drawPath(mPath, mPaint);

}



Проверяем, удобно ли спускаться. Теория - это одно, а практика - это совсем другое.

**Эффекты**

Контурные эффекты используются для управления отрисовкой контура. Они чрезвычайно полезны для рисования контурных графических примитивов, но могут быть применены к любому объекту **Paint**, чтобы повлиять на способ отрисовки их очертаний.

Используя этот вид эффектов, вы можете менять внешний вид углов фигур и их очертание.

* **CornerPathEffect**. Позволяет сглаживать острые углы в форме графического примитива, заменяя их на закругленные.
* **DashPathEffect**. Вместо рисования сплошного контура можете использовать DashPathEffect для создания очертания, состоящего из ломаных линий (тире/точек). Есть возможность указать любой шаблон повторения сплошных/пустых отрезков.
* **DiscretePathEffect**. Делает то же самое, что и DashPathEffect, но добавляет элемент случайности. Указываются длина каждого отрезка и степень отклонения от оригинального контура.
* **PathDashPathEffect**. Позволяет определить новую фигуру (контур), чтобы использовать ее в виде отпечатка оригинального контура.
* **SumPathEffect**. Добавляет последовательность из двух эффектов, каждый из которых применяется к оригинальному контуру, после чего результаты смешиваются;
* **ComposePathEffect**. Использует первый эффект, затем к полученному результату добавляет второй.

Контурные эффекты, влияющие на форму объекта, который должен быть нарисован, изменяют и область, занимаемую им. Благодаря этому любые эффекты для закрашивания, применяемые к данной фигуре, отрисовываются в новых границах.

Контурные эффекты применяются к объекту **Paint** с помощью метода **setPathEffect()**

**DashPathEffect**

Сплошную линию можно сделать пунктирной с помощью класса **DashPathEffect**. Перепишем немного код.

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.DashPathEffect;

import android.graphics.Paint;

import android.graphics.Path;

import android.util.AttributeSet;

import android.view.View;

public class PathView extends View {

private Paint mPaint;

private Path mPath;

private DashPathEffect mDashPathEffect;

public PathView(Context context) {

super(context);

init();

}

public PathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mPaint = new Paint();

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setAntiAlias(true);

mPaint.setColor(Color.RED);

mPaint.setStrokeWidth(3);

float[] intervals = new float[] { 60.0f, 10.0f };

float phase = 0;

mPath = new Path();

mDashPathEffect = new DashPathEffect(intervals, phase);

mPaint.setPathEffect(mDashPathEffect);

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPath.moveTo(50, 50);

mPath.lineTo(150, 50);

mPath.lineTo(150, 100);

mPath.lineTo(250, 100);

mPath.lineTo(250, 150);

mPath.lineTo(350, 150);

mPath.lineTo(350, 200);

mPath.lineTo(450, 200);

canvas.drawPath(mPath, mPaint);

}

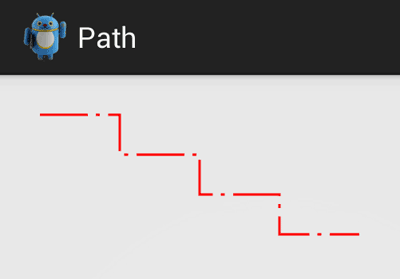
}

Результат.



Вам нужно указать длину отрезка для пунктира и длину отрезка для разрыва между двумя отрезками пунктира. Эта комбинация будет циклично использована для прорисовки всей линии. Пунктирная линия может быть сложной. Задайте массив для переменной **intervals**, чтобы увидеть разницу.

float[] intervals = new float[] { 60.0f, 10.0f, 5.0f, 10.5f };



**CornerPathEffect**

С помощью **CornerPathEffect** можно закруглить углы у прямых линий, чтобы ступеньки стали скользкими. Но проходимость коробки увеличится.

private void init() {

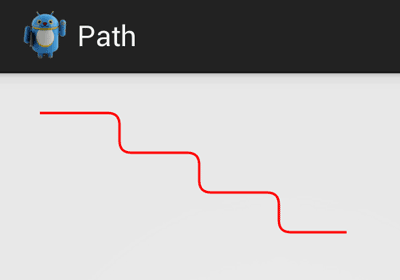
...

float radius = 15.0f;

mCornerPathEffect = new CornerPathEffect(radius);

mPaint.setPathEffect(mCornerPathEffect);

}



**PathDashPathEffect**

**PathDashPathEffect** позволяет определить новую фигуру, чтобы использовать её в виде отпечатка оригинального контура.

private void init() {

...

Path pathShape = new Path();

pathShape.addCircle(10, 10, 5, Direction.CCW);

float advance = 15.0f;

float phase = 10.0f;

PathDashPathEffect.Style style = PathDashPathEffect.Style.ROTATE;

mPathDashPathEffect = new PathDashPathEffect(

pathShape, advance, phase, style);

mPaint.setPathEffect(mPathDashPathEffect);

}



**Бегущие муравьи**

Вы все встречали эффект "бегущие муравьи" в графических редакторах. Применим его к объекту класса **PathDashPathEffect**, увеличивая смещение.

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Path;

import android.graphics.Path.Direction;

import android.graphics.PathDashPathEffect;

import android.util.AttributeSet;

import android.view.View;

public class PathView extends View {

private Paint mPaint;

private Path mPath;

private Path mShapePath;

private float mPhase;

private float mAdvance;

private PathDashPathEffect pathDashPathEffect;

private PathDashPathEffect.Style mStyle;

public PathView(Context context) {

super(context);

init();

}

public PathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mPaint = new Paint();

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setAntiAlias(true);

mPaint.setColor(Color.RED);

mPaint.setStrokeWidth(12);

mPath = new Path();

mShapePath = new Path();

mShapePath.addCircle(8, 8, 8, Direction.CCW);

mPhase = 0;

mAdvance = 30.0f;

mStyle = PathDashPathEffect.Style.ROTATE;

pathDashPathEffect = new PathDashPathEffect(

mShapePath, mAdvance, mPhase, mStyle);

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPath.reset();

mPath.moveTo(50, 50);

mPath.lineTo(50, getHeight() - 50);

mPath.lineTo(getWidth() - 50, getHeight() - 50);

mPath.lineTo(getWidth() - 50, 50);

mPath.close();

mPhase++;

pathDashPathEffect = new PathDashPathEffect(

mShapePath, mAdvance, mPhase, mStyle);

mPaint.setPathEffect(pathDashPathEffect);

canvas.drawPath(mPath, mPaint);

invalidate();

}

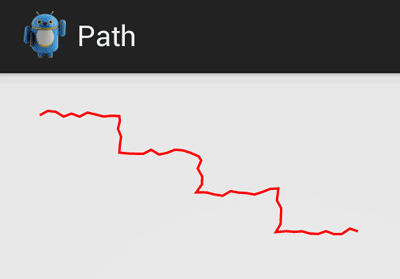
}

На странице [Effect of advance, phase, style in PathDashPathEffect](http://android-coding.blogspot.ru/2014/05/effect-of-advance-phase-style-in.html) автор примера поиграл с параметрами.

**DiscretePathEffect**

**DiscretePathEffect** позволяет "сломать" прямую линию, чтобы получить ломаную с элементом случайности. Полученная ломанная линия будет состоять из отдельных отрезков. Мы можем воздействовать на длину и степень излома.

Показать код (ткните лапой)

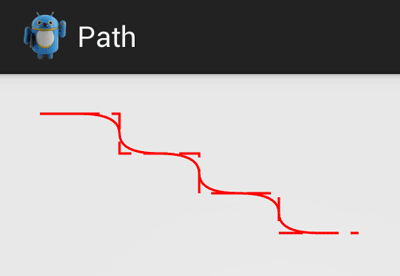


**SumPathEffect**

**SumPathEffect** добавляет последовательность из двух эффектов, каждый из которых применяется к оригинальному контуру, после чего результаты смешиваются. По сути, два эффекта накладываются друг на друга.

Суммируем эффекты **CornerPathEffect** и **DashPathEffect**. Для наглядности я чуть изменил параметры у эффектов, чтобы было виден результат наложения двух эффектов на лестницу - вы должны увидеть две линии - прерывистую и скруглённую.

Показать код (щелкните мышкой)



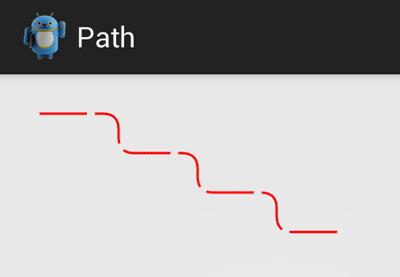
**ComposePathEffect**

**ComposePathEffect** использует первый эффект, затем к полученному результату добавляет второй. Таким образом, мы можем сделать нашу лестницу скруглённой, а затем прерывистой. Порядок эффектов имеет значение, хотя в нашем примере это не принципиально.

Заменим класс **SumPathEffect** на **ComposePathEffect** из предыдущего примера и посмотрим на результат.

mComposePathEffect = new ComposePathEffect(mDashPathEffect, mCornerPathEffect);

mPaint.setPathEffect(mComposePathEffect);



**Пример**

В документации есть отдельный пример на эту тему. При запуске примера мы увидим шесть вариантов эффектов, причём четыре из них будут анимированными! Поэтому желательно запустить проект и взглянуть на пример в действии, так как картинка не передаст прелесть эффектов.

package ru.alexanderklimov.patheffect;

import android.os.Bundle;

import android.app.Activity;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.ComposePathEffect;

import android.graphics.CornerPathEffect;

import android.graphics.DashPathEffect;

import android.graphics.PathDashPathEffect;

import android.graphics.Paint;

import android.graphics.Path;

import android.graphics.PathEffect;

import android.graphics.RectF;

import android.view.KeyEvent;

import android.view.View;

public class MainActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// setContentView(R.layout.activity\_main);

setContentView(new EffectView(this));

}

private static class EffectView extends View {

private Paint mPaint;

private Path mPath;

private PathEffect[] mEffects;

private int[] mColors;

private float mPhase;

private static void makeEffects(PathEffect[] e, float phase) {

e[0] = null; // без эффектов. просто ломаная линия

e[1] = new CornerPathEffect(10); // сглаживание острых углов у ломаной

// анимированные эффекты

e[2] = new DashPathEffect(new float[] {10, 5, 5, 5}, phase);

e[3] = new PathDashPathEffect(makePathDash(), 12, phase,

PathDashPathEffect.Style.ROTATE);

e[4] = new ComposePathEffect(e[2], e[1]);

e[5] = new ComposePathEffect(e[3], e[1]);

}

public EffectView(Context context) {

super(context);

setFocusable(true);

setFocusableInTouchMode(true);

mPaint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(6);

mPath = makeFollowPath();

mEffects = new PathEffect[6];

mColors = new int[] { Color.BLACK, Color.RED, Color.BLUE,

Color.GREEN, Color.MAGENTA, Color.BLACK

};

}

@Override protected void onDraw(Canvas canvas) {

canvas.drawColor(Color.WHITE);

RectF bounds = new RectF();

mPath.computeBounds(bounds, false);

canvas.translate(10 - bounds.left, 10 - bounds.top);

makeEffects(mEffects, mPhase);

mPhase += 1;

invalidate();

for (int i = 0; i < mEffects.length; i++) {

mPaint.setPathEffect(mEffects[i]);

mPaint.setColor(mColors[i]);

canvas.drawPath(mPath, mPaint);

canvas.translate(0, 28);

}

}

@Override public boolean onKeyDown(int keyCode, KeyEvent event) {

switch (keyCode) {

case KeyEvent.KEYCODE\_DPAD\_CENTER:

mPath = makeFollowPath();

return true;

}

return super.onKeyDown(keyCode, event);

}

private static Path makeFollowPath() {

Path p = new Path();

for (int i = 1; i <= 15; i++) {

p.lineTo(i\*20, (float)Math.random() \* 35);

}

return p;

}

private static Path makePathDash() {

Path p = new Path();

p.moveTo(4, 0);

p.lineTo(0, -4);

p.lineTo(8, -4);

p.lineTo(12, 0);

p.lineTo(8, 4);

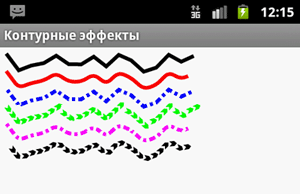
p.lineTo(0, 4);

return p;

}

}

}



**Примеры с контурами**

Продолжим опыты с контурами. Подготовим новый класс **Shape**, который будет отвечать за фигуры.

package ru.alexanderklimov.path;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Path;

public class Shape {

private Paint mPaint;

private Path mPath;

public Shape() {

mPaint = new Paint();

mPaint.setColor(Color.BLUE);

mPaint.setStrokeWidth(3);

mPaint.setStyle(Paint.Style.STROKE);

mPath = new Path();

}

public void setCircle(float x, float y, float radius, Path.Direction dir) {

mPath.reset();

mPath.addCircle(x, y, radius, dir);

}

public Path getPath() {

return mPath;

}

public Paint getPaint() {

return mPaint;

}

}

В класс **PathView** внесём изменения.

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Path.Direction;

import android.util.AttributeSet;

import android.view.View;

public class PathView extends View {

private Shape mShape;

private float mRatioRadius;

public PathView(Context context) {

super(context);

init();

}

public PathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mShape = new Shape();

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

int width = getWidth();

int height = getHeight();

if ((width == 0) || (height == 0)) {

return;

}

float x = (float) width / 2.0f;

float y = (float) height / 2.0f;

float radius;

if (width > height) {

radius = height \* mRatioRadius;

} else {

radius = width \* mRatioRadius;

}

mShape.setCircle(x, y, radius, Direction.CCW);

canvas.drawPath(mShape.getPath(), mShape.getPaint());

}

public void setShapeRadiusRatio(float ratio) {

mRatioRadius = ratio;

}

}

Добавим в разметку компонент **SeekBar**, чтобы динамически менять размер контура.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/LinearLayout1"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

tools:context="${relativePackage}.${activityClass}" >

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="radius(%)" />

<SeekBar

android:id="@+id/seekBar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="100"

android:progress="50" />

<ru.alexanderklimov.path.PathView

android:id="@+id/pathView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_marginTop="164dp" />

</LinearLayout>

Код активности.

package ru.alexanderklimov.path;

import android.app.Activity;

import android.os.Bundle;

import android.widget.SeekBar;

import android.widget.SeekBar.OnSeekBarChangeListener;

public class MainActivity extends Activity {

private SeekBar mSeekBar;

private PathView mPathView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mSeekBar = (SeekBar) findViewById(R.id.seekBar);

mPathView = (PathView) findViewById(R.id.pathView);

float defaultRatio = (float) (mSeekBar.getProgress())

/ (float) (mSeekBar.getMax());

mPathView.setShapeRadiusRatio(defaultRatio);

mSeekBar.setOnSeekBarChangeListener(onSeekBarChangeListener);

};

OnSeekBarChangeListener onSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

float defaultRatio = (float) (mSeekBar.getProgress())

/ (float) (mSeekBar.getMax());

mPathView.setShapeRadiusRatio(defaultRatio);

mPathView.invalidate();

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

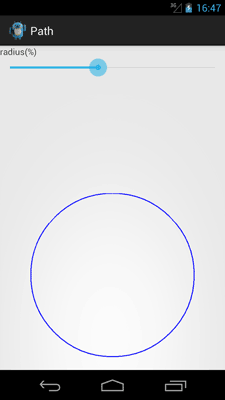
public void onStopTrackingTouch(SeekBar seekBar) {

}

};

}

Запустив проект, вы можете с помощью ползунка менять размеры контура, который в данной реализации является окружностью.



Усложним пример. Будем использовать не только окружность, но и другие фигуры. Добавим в класс **Shape** метод **setPolygon()**:

public void setPolygon(float x, float y, float radius, int numOfPt) {

double section = 2.0 \* Math.PI / numOfPt;

mPath.reset();

mPath.moveTo((float) (x + radius \* Math.cos(0)), (float) (y + radius

\* Math.sin(0)));

for (int i = 1; i < numOfPt; i++) {

mPath.lineTo((float) (x + radius \* Math.cos(section \* i)),

(float) (y + radius \* Math.sin(section \* i)));

}

mPath.close();

}

Класс **PathView** потребует небольшой переделки.

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Canvas;

import android.util.AttributeSet;

import android.view.View;

public class PathView extends View {

private Shape mShape;

private float mRatioRadius;

private int mNumberOfPoint = 3; // default

public PathView(Context context) {

super(context);

init();

}

public PathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mShape = new Shape();

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

int width = getWidth();

int height = getHeight();

if ((width == 0) || (height == 0)) {

return;

}

float x = (float) width / 2.0f;

float y = (float) height / 2.0f;

float radius;

if (width > height) {

radius = height \* mRatioRadius;

} else {

radius = width \* mRatioRadius;

}

mShape.setPolygon(x, y, radius, mNumberOfPoint);

canvas.drawPath(mShape.getPath(), mShape.getPaint());

}

public void setShapeRadiusRatio(float ratio) {

mRatioRadius = ratio;

}

public void setNumberOfPoint(int pt) {

mNumberOfPoint = pt;

}

}

В разметке до компонента PathView добавьте пару новых компонентов.

<TextView

android:id="@+id/textViewPoint"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Number of point in polygon: 3" />

<SeekBar

android:id="@+id/seekBarPoint"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="10"

android:progress="0" />

Код для активности.

package ru.alexanderklimov.path;

import android.app.Activity;

import android.os.Bundle;

import android.widget.SeekBar;

import android.widget.SeekBar.OnSeekBarChangeListener;

import android.widget.TextView;

public class MainActivity extends Activity {

private SeekBar mSeekBar;

private PathView mPathView;

private SeekBar mPointSeekBar;

private TextView mPointTextView;

final static int MIN\_PT = 3;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mSeekBar = (SeekBar) findViewById(R.id.seekBar);

mPathView = (PathView) findViewById(R.id.pathView);

float defaultRatio = (float) (mSeekBar.getProgress())

/ (float) (mSeekBar.getMax());

mPathView.setShapeRadiusRatio(defaultRatio);

mSeekBar.setOnSeekBarChangeListener(onSeekBarChangeListener);

mPointTextView = (TextView)findViewById(R.id.textViewPoint);

mPointSeekBar = (SeekBar)findViewById(R.id.seekBarPoint);

mPointSeekBar.setOnSeekBarChangeListener(onPointSeekBarChangeListener);

}

OnSeekBarChangeListener onSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

float defaultRatio = (float) (mSeekBar.getProgress())

/ (float) (mSeekBar.getMax());

mPathView.setShapeRadiusRatio(defaultRatio);

mPathView.invalidate();

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

};

OnSeekBarChangeListener onPointSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

// TODO Auto-generated method stub

int pt = progress + MIN\_PT;

mPointTextView.setText("Number of point in polygon: " + String.valueOf(pt));

mPathView.setNumberOfPoint(pt);

mPathView.invalidate();

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

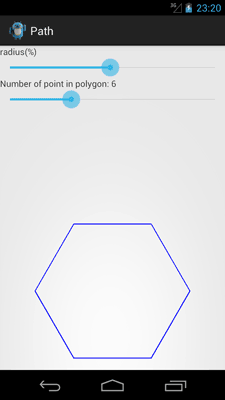
// TODO Auto-generated method stub

}

};

}

Теперь мы можем создавать более сложные фигуры - многоугольники, начиная с треугольника, затем четырёхугольник, пятиугольники и так далее.



Следующий этап - создание звёзд, пятиконечной, шестиконечной и т.д.

Опять добавим код в класс **Shape**.

package ru.alexanderklimov.path;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Path;

public class Shape {

private Paint mPaint;

private Path mPath;

public Shape() {

mPaint = new Paint();

mPaint.setColor(Color.BLUE);

mPaint.setStrokeWidth(3);

mPaint.setStyle(Paint.Style.STROKE);

mPath = new Path();

}

public void setCircle(float x, float y, float radius, Path.Direction dir) {

mPath.reset();

mPath.addCircle(x, y, radius, dir);

}

public void setPolygon(float x, float y, float radius, int numOfPt) {

double section = 2.0 \* Math.PI / numOfPt;

mPath.reset();

mPath.moveTo((float) (x + radius \* Math.cos(0)), (float) (y + radius

\* Math.sin(0)));

for (int i = 1; i < numOfPt; i++) {

mPath.lineTo((float) (x + radius \* Math.cos(section \* i)),

(float) (y + radius \* Math.sin(section \* i)));

}

mPath.close();

}

public void setStar(float x, float y, float radius, float innerRadius,

int numOfPt) {

double section = 2.0 \* Math.PI / numOfPt;

mPath.reset();

mPath.moveTo((float) (x + radius \* Math.cos(0)), (float) (y + radius

\* Math.sin(0)));

mPath.lineTo((float) (x + innerRadius \* Math.cos(0 + section / 2.0)),

(float) (y + innerRadius \* Math.sin(0 + section / 2.0)));

for (int i = 1; i < numOfPt; i++) {

mPath.lineTo((float) (x + radius \* Math.cos(section \* i)),

(float) (y + radius \* Math.sin(section \* i)));

mPath.lineTo(

(float) (x + innerRadius

\* Math.cos(section \* i + section / 2.0)),

(float) (y + innerRadius

\* Math.sin(section \* i + section / 2.0)));

}

mPath.close();

}

public Path getPath() {

return mPath;

}

public Paint getPaint() {

return mPaint;

}

}

Внесём изменения в класс **PathView**.

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Canvas;

import android.util.AttributeSet;

import android.view.View;

public class PathView extends View {

private Shape mShape;

private float mRatioRadius;

private float mRatioInnerRadius;

private int mNumberOfPoint = 3; // default

public PathView(Context context) {

super(context);

init();

}

public PathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mShape = new Shape();

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

int width = getWidth();

int height = getHeight();

if ((width == 0) || (height == 0)) {

return;

}

float x = (float) width / 2.0f;

float y = (float) height / 2.0f;

float radius;

float innerRadius;

if (width > height) {

radius = height \* mRatioRadius;

innerRadius = height \* mRatioInnerRadius;

} else {

radius = width \* mRatioRadius;

innerRadius = width \* mRatioInnerRadius;

}

mShape.setStar(x, y, radius, innerRadius, mNumberOfPoint);

canvas.drawPath(mShape.getPath(), mShape.getPaint());

}

public void setShapeRadiusRatio(float ratio) {

mRatioRadius = ratio;

}

public void setShapeInnerRadiusRatio(float ratio) {

mRatioInnerRadius = ratio;

}

public void setNumberOfPoint(int pt) {

mNumberOfPoint = pt;

}

}

Разметка.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/LinearLayout1"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

tools:context="${relativePackage}.${activityClass}" >

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Radius(%)" />

<SeekBar

android:id="@+id/seekBarRadius"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="100"

android:progress="50" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Inner radius(%)" />

<SeekBar

android:id="@+id/seekBarInnerRadius"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="100"

android:progress="25" />

<TextView

android:id="@+id/textViewPoint"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Number of point in polygon: 3" />

<SeekBar

android:id="@+id/seekBarPoint"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="10"

android:progress="0" />

<ru.alexanderklimov.path.PathView

android:id="@+id/pathView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_marginTop="164dp" />

</LinearLayout>

Код активности.

package ru.alexanderklimov.path;

import android.app.Activity;

import android.os.Bundle;

import android.widget.SeekBar;

import android.widget.SeekBar.OnSeekBarChangeListener;

import android.widget.TextView;

public class MainActivity extends Activity {

final static int MIN\_PT = 3;

private SeekBar mRadiusSeekBar;

private SeekBar mInnerRadiusSeekBar;

private SeekBar mPointSeekBar;

private TextView mPointTextView;

private PathView mPathView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mRadiusSeekBar = (SeekBar) findViewById(R.id.seekBarRadius);

mInnerRadiusSeekBar = (SeekBar) findViewById(R.id.seekBarInnerRadius);

mPointSeekBar = (SeekBar) findViewById(R.id.seekBarPoint);

mPointTextView = (TextView) findViewById(R.id.textViewPoint);

mPathView = (PathView) findViewById(R.id.pathView);

// float defaultRatio = (float) (mRadiusSeekBar.getProgress())

// / (float) (mRadiusSeekBar.getMax());

float defaultInnerRatio = (float) (mInnerRadiusSeekBar.getProgress())

/ (float) (mInnerRadiusSeekBar.getMax());

// mPathView.setShapeRadiusRatio(defaultRatio);

mPathView.setShapeInnerRadiusRatio(defaultInnerRatio);

mRadiusSeekBar.setOnSeekBarChangeListener(onSeekBarChangeListener);

mInnerRadiusSeekBar

.setOnSeekBarChangeListener(onInnerRadiusSeekBarChangeListener);

mPointSeekBar.setOnSeekBarChangeListener(onPointSeekBarChangeListener);

}

OnSeekBarChangeListener onSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

float defaultRatio = (float) (mRadiusSeekBar.getProgress())

/ (float) (mRadiusSeekBar.getMax());

mPathView.setShapeRadiusRatio(defaultRatio);

mPathView.invalidate();

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

};

OnSeekBarChangeListener onPointSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

// TODO Auto-generated method stub

int pt = progress + MIN\_PT;

mPointTextView.setText("Number of point in polygon: "

+ String.valueOf(pt));

mPathView.setNumberOfPoint(pt);

mPathView.invalidate();

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

};

OnSeekBarChangeListener onInnerRadiusSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

// TODO Auto-generated method stub

float ratio = (float) (mInnerRadiusSeekBar.getProgress())

/ (float) (mInnerRadiusSeekBar.getMax());

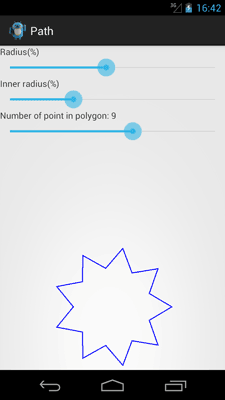
mPathView.setShapeInnerRadiusRatio(ratio);

mPathView.invalidate();

}

};

}

Чтобы закруглить углы у звёзд, применим эффект **CornerPathEffect**. Добавим код в конструктор класса **Shape**.

public Shape() {

mPaint = new Paint();

mPaint.setColor(Color.BLUE);

mPaint.setStrokeWidth(3);

mPaint.setStyle(Paint.Style.STROKE);

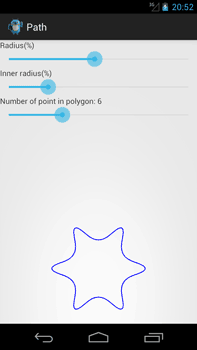
float radius = 50.0f;

CornerPathEffect cornerPathEffect = new CornerPathEffect(radius);

mPaint.setPathEffect(cornerPathEffect);

mPath = new Path();

}



Чтобы залить фигуру цветом, нужно использовать вместо стиля **Paint.Style.STROKE** стиль **Paint.Style.FILL** или **Paint.Style.FILL\_AND\_STROKE**.

mPaint.setStyle(Paint.Style.FILL\_AND\_STROKE);



**Вращение**

Чтобы вращать контур, нужно создать объект класса **Matrix** и вызвать метод **postRotate()**.

Добавим в класс **PathView** две новых переменных.

private float mRotate;

private Matrix mMatrix;

Добавим строчку кода в метод init():

private void init() {

mShape = new Shape();

mMatrix = new Matrix();

}

Добавим код в onDraw():

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

...

mShape.setStar(x, y, radius, innerRadius, mNumberOfPoint);

// Rotate the path by angle in degree

Path path = mShape.getPath();

mMatrix.reset();

mMatrix.postRotate(mRotate, x, y);

path.transform(mMatrix);

canvas.drawPath(mShape.getPath(), mShape.getPaint());

}

Добавим новый метод.

public void setShapeRotate(int degree) {

mRotate = (float) degree;

}

Добавим в разметку активности ещё один SeekBar

...

<TextView

android:id="@+id/textViewRotate"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="rotate :"/>

<SeekBar

android:id="@+id/seekBarRotate"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="360"

android:progress="180" />

<ru.alexanderklimov.path.PathView

android:id="@+id/pathView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_marginTop="164dp" />

И добавляем код в класс активности:

// Переменные класса

private SeekBar mRotateSeekBar;

private TextView mRotateTextView;

// для метода onCreate()

mRotateTextView = (TextView) findViewById(R.id.textViewRotate);

mRotateSeekBar = (SeekBar) findViewById(R.id.seekBarRotate);

mRotateSeekBar

.setOnSeekBarChangeListener(onRotateSeekBarChangeListener);

mPathView.setRotation(0); // set default rotate degree

// слушатель

OnSeekBarChangeListener onRotateSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

// TODO Auto-generated method stub

int degree = progress - 180;

mRotateTextView.setText("Rotate : " + degree + " degree");

mPathView.setShapeRotate(degree);

mPathView.invalidate();

}

};

Вращать можно не только сам контур, но и холст вместо него. Эффект будет такой же, а по потреблению ресурсов даже может оказаться эффективнее. Закоментируем использование класса **Matrix** и добавим вызов метода **Canvas.rotate()** в методе **onDraw()** класса **PathView**.

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

int width = getWidth();

int height = getHeight();

if ((width == 0) || (height == 0)) {

return;

}

float x = (float) width / 2.0f;

float y = (float) height / 2.0f;

float radius;

float innerRadius;

if (width > height) {

radius = height \* mRatioRadius;

innerRadius = height \* mRatioInnerRadius;

} else {

radius = width \* mRatioRadius;

innerRadius = width \* mRatioInnerRadius;

}

mShape.setStar(x, y, radius, innerRadius, mNumberOfPoint);

// Rotate the path by angle in degree

// Path path = mShape.getPath();

// mMatrix.reset();

// mMatrix.postRotate(mRotate, x, y);

// path.transform(mMatrix);

// Save and rotate canvas

canvas.save();

canvas.rotate(mRotate, x, y);

canvas.drawPath(mShape.getPath(), mShape.getPaint());

// restore canvas

canvas.restore();

}

Теперь создадим эффект "бегущих муравьёв" при помощи **PathDashPathEffect**:

private Path mDashPath;

private float mPhase;

private void init() {

mShape = new Shape();

mDashPath = new Path();

mDashPath.addCircle(0, 0, 3, Direction.CCW);

mPhase = 0.0f;

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

...

mShape.setStar(x, y, radius, innerRadius, mNumberOfPoint);

// Save and rotate canvas

canvas.save();

canvas.rotate(mRotate, x, y);

**mPhase++;**

**PathDashPathEffect pathDashPathEffect = new PathDashPathEffect(**

**mDashPath, 15.0f, mPhase, PathDashPathEffect.Style.MORPH);**

**Paint paintDash = mShape.getPaint();**

**paintDash.setPathEffect(pathDashPathEffect);**

canvas.drawPath(mShape.getPath(), mShape.getPaint());

// restore canvas

canvas.restore();

**invalidate();**

}

**Использованные материалы**

[Android-er: Draw Path on canvas of custom View](http://android-er.blogspot.ru/2014/05/draw-path-on-canvas-of-custom-view.html)

[Android-er: Draw Path of polygon on canvas of custom View](http://android-er.blogspot.ru/2014/05/draw-path-of-polygon-on-canvas-of.html)

[Android-er: Draw star on canvas](http://android-er.blogspot.ru/2014/05/draw-star-on-canvas.html)

[Android-er: Rotate Path with Matrix](http://android-er.blogspot.ru/2014/06/rotate-path-with-matrix.html)

[Android-er: Draw rotated path by rotating canvas](http://android-er.blogspot.ru/2014/06/draw-rotated-path-by-rotating-canvas.html)

[Android-er: Implement running dash path, using PathDashPathEffect](http://android-er.blogspot.ru/2014/06/implement-running-dash-path-using.html)

[Android-er: Implement arrow-like PathDashPathEffect](http://android-er.blogspot.ru/2014/06/implement-arrow-like-pathdashpatheffect.html)