**[Урок 151. Рисование. PathEffect](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/348-urok-151-risovanie-patheffect.html)**

В этом уроке:

- рассматриваем PathEffect-объекты

У класса [PathEffect](http://developer.android.com/reference/android/graphics/PathEffect.html" \t "_blank) есть несколько наследников, которые позволяют влиять на рисуемые нами объекты. Рассмотрим на примерах их использование.

Создадим проект:

**Project name**: P1511\_PathEffect  
**Build Target**: Android 2.3.3   
**Application name**: PathEffect  
**Package name**: ru.startandroid.develop.p1511patheffect  
**Create Activity**: MainActivity

### CornerPathEffect

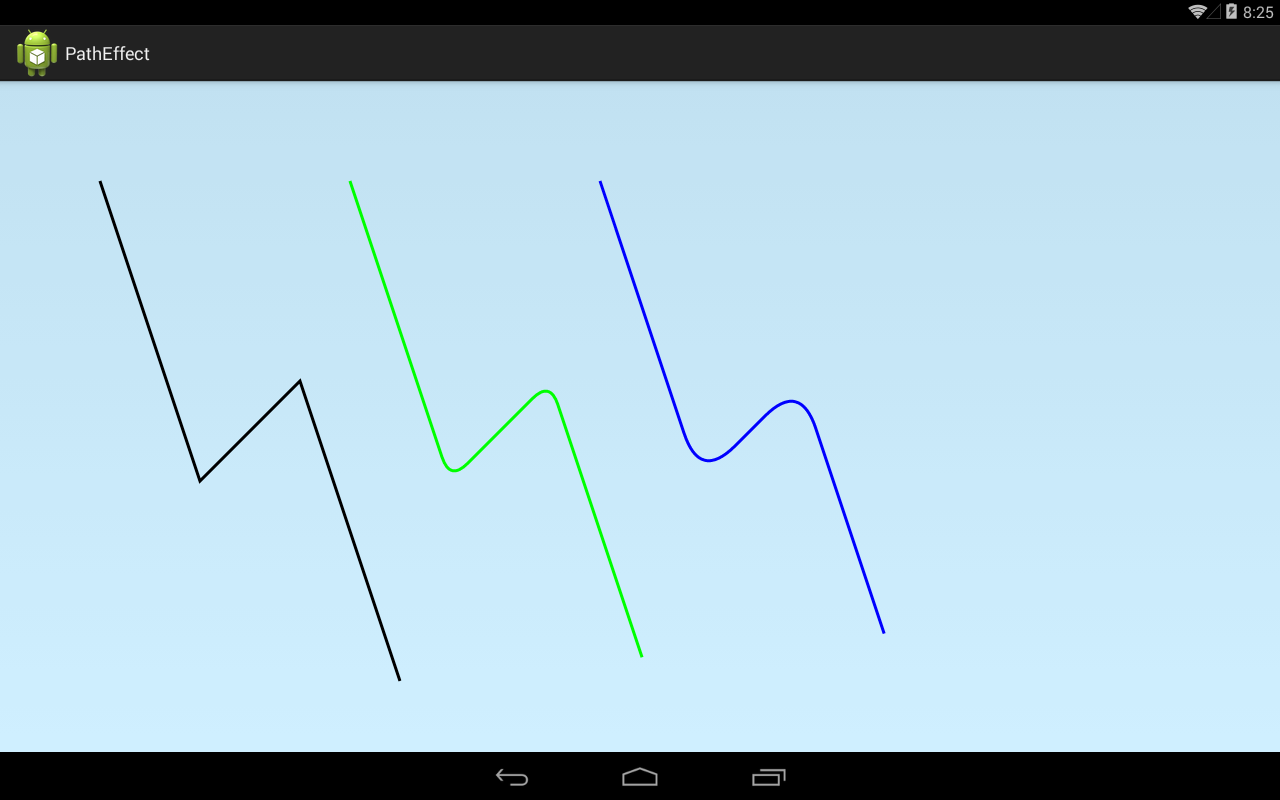
Эффект [CornerPathEffect](http://developer.android.com/reference/android/graphics/CornerPathEffect.html" \t "_blank) просто закругляет углы. На вход принимает радиус закругления.

Пишем в **MainActivity.java**:

**package** ru.startandroid.develop.p1511patheffect;  
  
**import** android.app.Activity;  
**import** android.content.Context;  
**import** android.graphics.Canvas;  
**import** android.graphics.Color;  
**import** android.graphics.CornerPathEffect;  
**import** android.graphics.Paint;  
**import** android.graphics.Path;  
**import** android.os.Bundle;  
**import** android.view.View;  
  
**public class** MainActivity **extends** Activity {  
  
  @Override  
  **protected void** onCreate(Bundle savedInstanceState) {  
    **super**.onCreate(savedInstanceState);  
    setContentView(**new** DrawView(**this**));  
  }  
  
  **class** DrawView **extends** View {  
  
    Path path;  
    Paint p1;  
    Paint p2;  
    Paint p3;  
  
    **public** DrawView(Context context) {  
      **super**(context);  
      path = **new** Path();  
      path.rLineTo(100, 300);  
      path.rLineTo(100, -100);  
      path.rLineTo(100, 300);  
  
      p1 = **new** Paint(Paint.ANTI\_ALIAS\_FLAG);  
      p1.setStyle(Paint.Style.STROKE);  
      p1.setStrokeWidth(3);  
  
      p2 = **new** Paint(p1);  
      p2.setColor(Color.GREEN);  
      p2.setPathEffect(**new** CornerPathEffect(25));  
  
      p3 = **new** Paint(p1);  
      p3.setColor(Color.BLUE);  
      p3.setPathEffect(**new** CornerPathEffect(50));  
  
    }  
  
    @Override  
    **protected void** onDraw(Canvas canvas) {  
      canvas.drawARGB(80, 102, 204, 255);  
  
      canvas.translate(100, 100);  
      canvas.drawPath(path, p1);  
  
      canvas.translate(250, 0);  
      canvas.drawPath(path, p2);  
  
      canvas.translate(250, 0);  
      canvas.drawPath(path, p3);  
  
    }  
  
  }  
  
}

Смотрим код. Мы создаем Path, состоящий из трех линий. Создаем три кисти с разными цветами: черную без эффектов, зеленую с закруглением с радиусом в 25, и синюю с закруглением 50. И рисуем фигуру три раза.

Результат:



### DiscretePathEffect

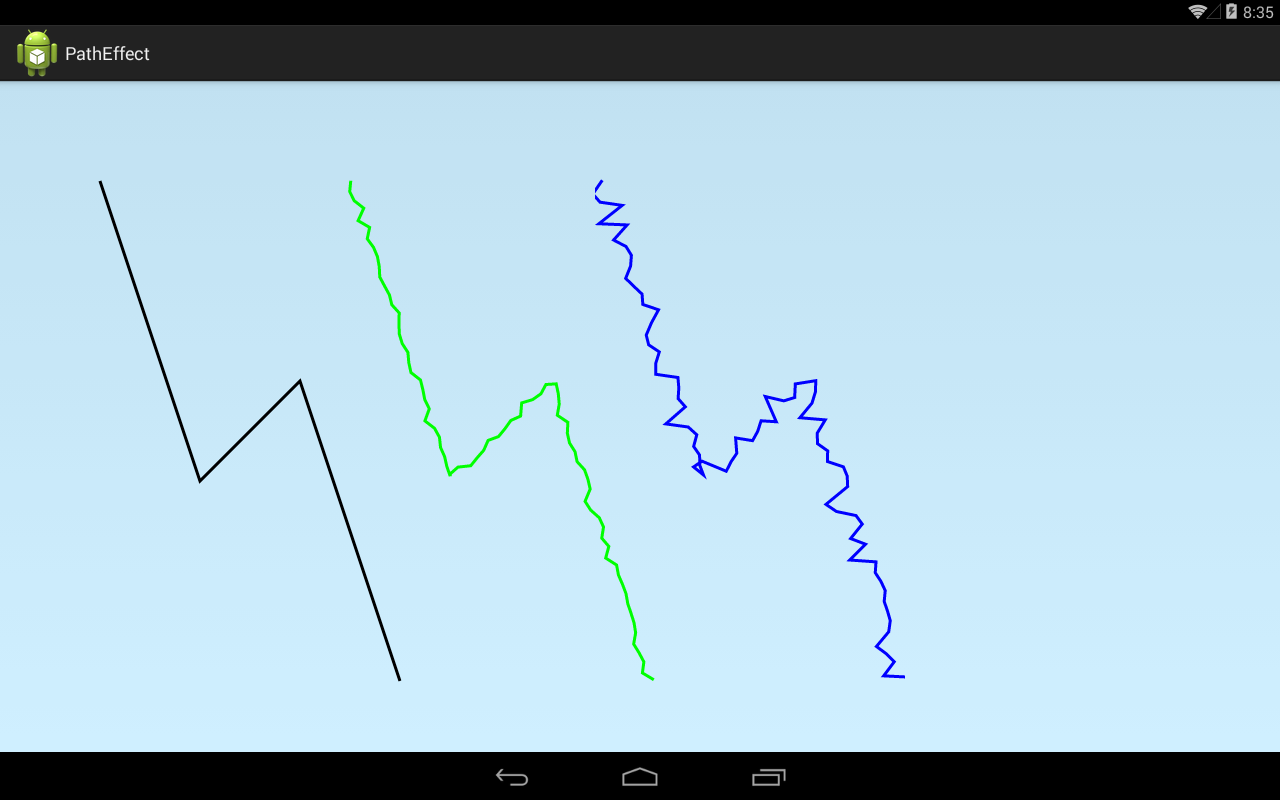
[DiscretePathEffect](http://developer.android.com/reference/android/graphics/DiscretePathEffect.html) позволяет получить ломанную линию из прямой. Полученная ломанная линия будет состоять из фрагментов, а мы можем повлиять на длину этих фрагментов (первый параметр конструктора) и степень излома (второй параметр).

Перепишем класс **DrawView**:

**class** DrawView **extends** View {  
        
      Path path;  
      Paint p1;  
      Paint p2;  
      Paint p3;  
        
      **public** DrawView(Context context) {  
        **super**(context);  
        path = **new** Path();  
        path.rLineTo(100, 300);  
        path.rLineTo(100, -100);  
        path.rLineTo(100, 300);  
          
        p1 = **new** Paint(Paint.ANTI\_ALIAS\_FLAG);  
        p1.setStyle(Paint.Style.STROKE);  
        p1.setStrokeWidth(3);  
          
        p2 = **new** Paint(p1);  
        p2.setColor(Color.GREEN);  
        p2.setPathEffect(**new** DiscretePathEffect(10,5));  
          
        p3 = **new** Paint(p1);  
        p3.setColor(Color.BLUE);  
        p3.setPathEffect(**new** DiscretePathEffect(10,15));  
          
      }  
        
      @Override  
      **protected void** onDraw(Canvas canvas) {  
        canvas.drawARGB(80, 102, 204, 255);  
          
        canvas.translate(100, 100);  
        canvas.drawPath(path, p1);  
          
        canvas.translate(250, 0);  
        canvas.drawPath(path, p2);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p3);  
          
      }  
        
    }

Для зеленой линии используем степень излома – 5, а для синей – 15. Длина фрагментов = 10.

Результат:



### DashPathEffect

С помощью [DashPathEffect](http://developer.android.com/reference/android/graphics/DashPathEffect.html" \t "_blank) мы из сплошной линии можем получить прерывистую. От нас требуется задать длину участка который будет прорисован и длину участка, который прорисован не будет, т.е. «пусто». Далее эта комбинация будет циклично использована для прорисовки всей линии.

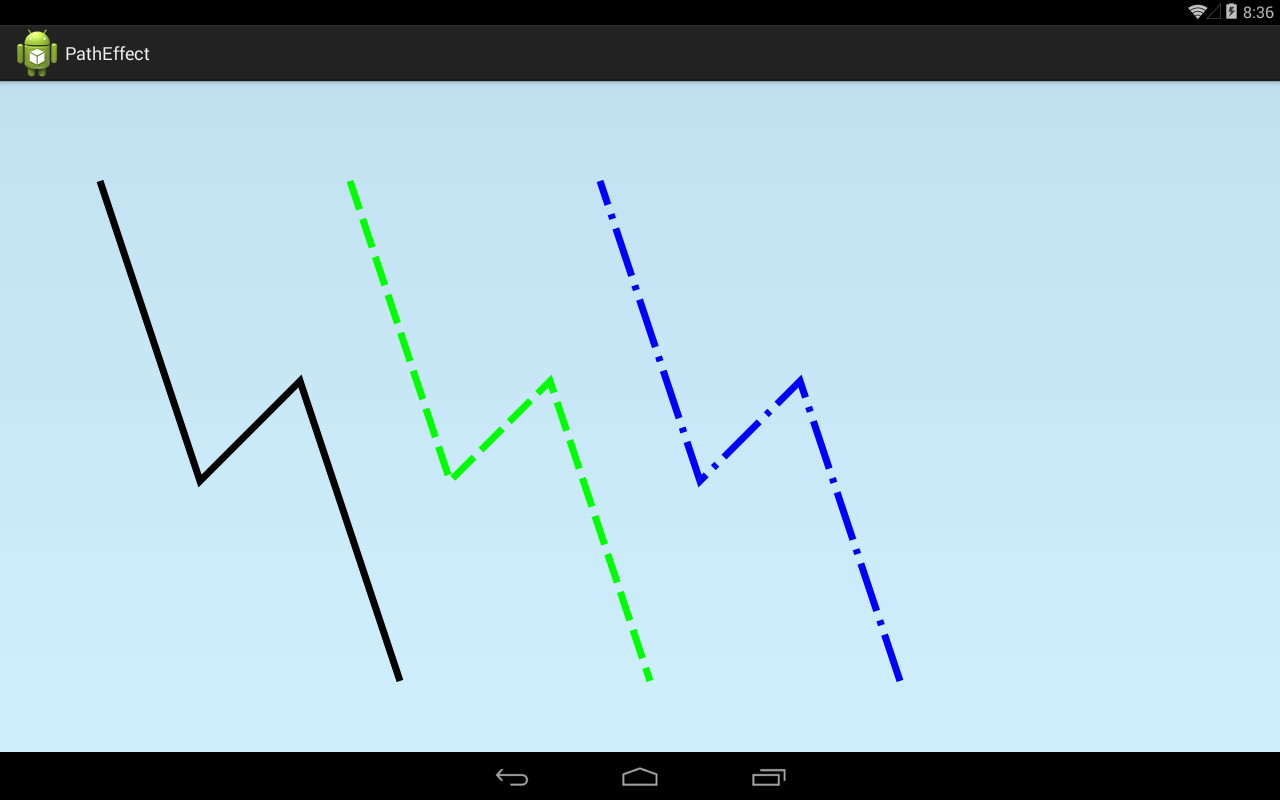
Перепишем класс **DrawView**:

**class** DrawView **extends** View {  
        
      Path path;  
      Paint p1;  
      Paint p2;  
      Paint p3;  
        
      **public** DrawView(Context context) {  
        **super**(context);  
        path = **new** Path();  
        path.rLineTo(100, 300);  
        path.rLineTo(100, -100);  
        path.rLineTo(100, 300);  
          
        p1 = **new** Paint(Paint.ANTI\_ALIAS\_FLAG);  
        p1.setStyle(Paint.Style.STROKE);  
        p1.setStrokeWidth(7);  
          
        p2 = **new** Paint(p1);  
        p2.setColor(Color.GREEN);  
        p2.setPathEffect(**new** DashPathEffect(**new float**[] { 30, 10}, 0));  
          
        p3 = **new** Paint(p1);  
        p3.setColor(Color.BLUE);  
        p3.setPathEffect(**new** DashPathEffect(**new float**[] { 50, 10, 5, 10 }, 25));  
          
      }  
        
      @Override  
      **protected void** onDraw(Canvas canvas) {  
        canvas.drawARGB(80, 102, 204, 255);  
          
        canvas.translate(100, 100);  
        canvas.drawPath(path, p1);  
          
        canvas.translate(250, 0);  
        canvas.drawPath(path, p2);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p3);  
          
      }  
        
    }

Для зеленой линии мы настраиваем длину выводимого участка = 30, длину пустоты = 10. Мы помещаем эти значение в массив и передаем в DashPathEffect-конструктор первым параметром. Вторым параметром идет отступ, его не используем.

Для синей линии мы задаем чуть более сложную последовательность: 50 выводить, 10 пусто, 5, выводить, 10 пусто. Т.е. принцип наверно уже понятен. Система будет поочередно использовать значения из массива для определения длины рисуемого куска линии и длины следующей за ним пустоты. Отступ используем в 25.

Результат:



Зеленая линяя состоит из отрезков длиной 30 и пустоты длиной 10. А синяя из отрезка длиной 50, пустоты 10, отрезка 5, пустоты 10. У синей линии первый отрезок выглядит короче остальных больших. Это сработал отступ в 25. Если вы этот отступ повесите в цикл, то линия оживет и поедет на месте, что выглядит достаточно эффектно.

### PathDashPathEffect

[PathDashPathEffect](http://developer.android.com/reference/android/graphics/PathDashPathEffect.html) позволяет сделать пунктирную линию, но в качестве пунктира можно использовать свой Path-объект.

Перепишем класс **DrawView**:

**class** DrawView **extends** View {  
        
      Path path;  
      Path pathStamp;  
      Paint p1;  
      Paint p2;  
      Paint p3;  
      Paint p4;  
        
      **public** DrawView(Context context) {  
        **super**(context);  
        path = **new** Path();  
        path.addRect(-100, 0, 100, 500, Path.Direction.CW);  
          
        pathStamp = **new** Path();  
        pathStamp.lineTo(-10, -10);  
        pathStamp.lineTo(10, 0);  
        pathStamp.lineTo(-10, 10);  
        pathStamp.close();  
          
        p1 = **new** Paint(Paint.ANTI\_ALIAS\_FLAG);  
        p1.setStyle(Paint.Style.STROKE);  
        p1.setStrokeWidth(20);  
          
        p2 = **new** Paint(p1);  
        p2.setColor(Color.GREEN);  
        p2.setPathEffect(**new** PathDashPathEffect(pathStamp, 20, 0, PathDashPathEffect.Style.MORPH));  
          
        p3 = **new** Paint(p1);  
        p3.setColor(Color.BLUE);  
        p3.setPathEffect(**new** PathDashPathEffect(pathStamp, 20, 0, PathDashPathEffect.Style.ROTATE));  
          
        p4 = **new** Paint(p1);  
        p4.setColor(Color.RED);  
        p4.setPathEffect(**new** PathDashPathEffect(pathStamp, 20, 10, PathDashPathEffect.Style.TRANSLATE));  
          
      }  
        
      @Override  
      **protected void** onDraw(Canvas canvas) {  
        canvas.drawARGB(80, 102, 204, 255);  
          
        canvas.translate(120, 100);  
        canvas.drawPath(path, p1);  
          
        canvas.translate(250, 0);  
        canvas.drawPath(path, p2);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p3);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p4);  
      }  
        
    }

Создаем объект path с прямоугольником и pathStamp в виде стрелки. Далее на кисти вешаем эффект PathDashPathEffect. Его конструктор на вход принимает:

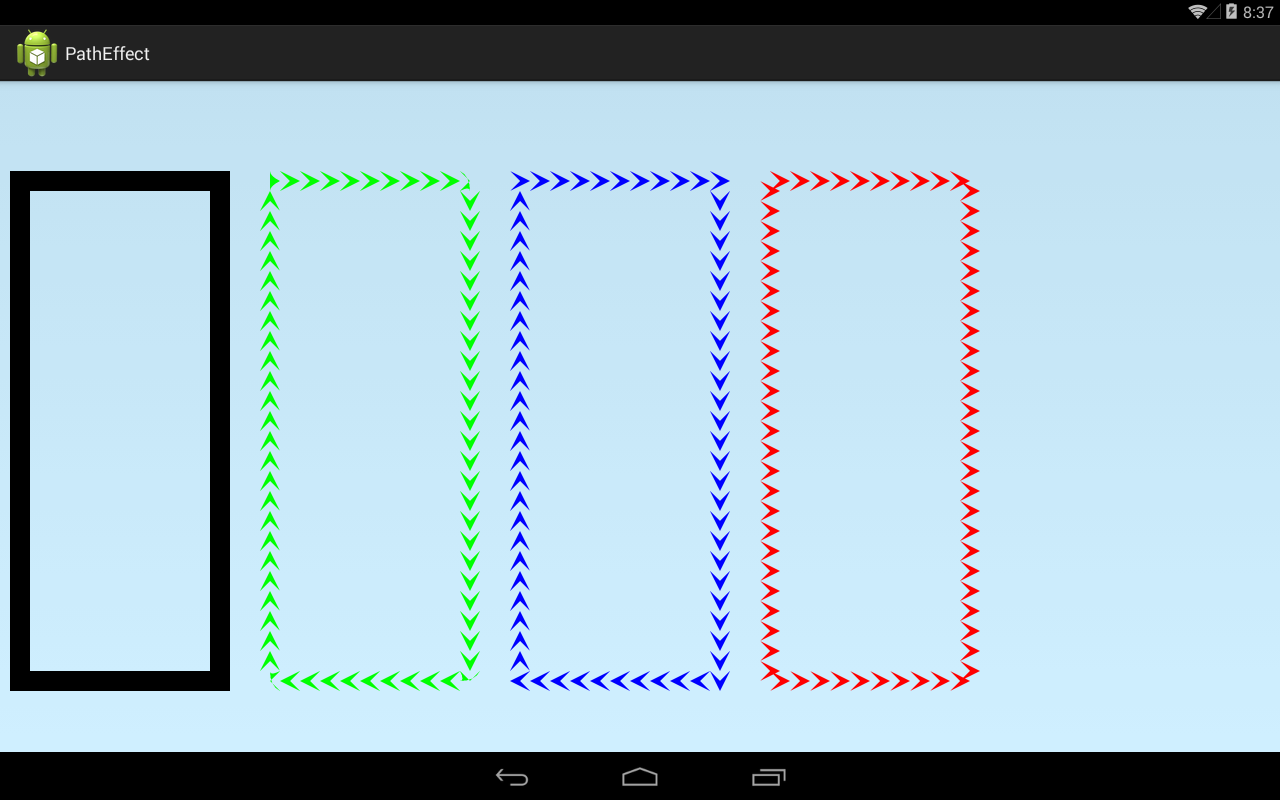
- path-объект, который будет использован в качестве пунктира

- расстояние между пунктирами

- отступ от начала

- стиль эффекта

Результат:



Линии состоят из стрелок (объект pathStamp). Расстояние между ними = 20. По стилям хелп не дает толковой инфы. По моим наблюдениям могу предположить следующее:

PathDashPathEffect.Style.MORPH – срезает пунктир на углах (видно по зеленой линии)

PathDashPathEffect.Style.ROTATE – корректно работает с углами (видно по синей линии)

PathDashPathEffect.Style.TRANSLATE – не поворачивает pathStamp по направлению основной линии (видно по красной фигуре)

Для красной линии я использовал небольшой отступ, это видно – стрелки идут не из самого угла. Опять же, повесив отступ в цикл вы получите ожившую линию.

### SumPathEffect и ComposePathEffect

Позволяют нам комбинировать два эффекта, которые подаются им на вход.

[ComposePathEffect](http://developer.android.com/reference/android/graphics/ComposePathEffect.html) применит сначала один эффект, потом к получившемуся результату – второй и выведет результат. [SumPathEffect](http://developer.android.com/reference/android/graphics/SumPathEffect.html" \t "_blank) – применит к искомой фигуре один эффект, выведет результат, затем применит к искомой фигуре второй эффект и выведет результат.

Перепишем класс **DrawView**:

**class** DrawView **extends** View {  
        
      Path path;  
      Paint p1;  
      Paint p2;  
      Paint p3;  
      Paint p4;  
      Paint p5;  
        
      **public** DrawView(Context context) {  
        **super**(context);  
        path = **new** Path();  
        path.addRect(-100, 0, 100, 500, Path.Direction.CW);  
        
        PathEffect pe1 = **new** CornerPathEffect(100);  
        PathEffect pe2 = **new** DashPathEffect(**new float**[] { 20, 5}, 0);  
        PathEffect pe3 = **new** ComposePathEffect(pe2, pe1);  
        PathEffect pe4 = **new** SumPathEffect(pe1, pe2);  
          
        p1 = **new** Paint(Paint.ANTI\_ALIAS\_FLAG);  
        p1.setStyle(Paint.Style.STROKE);  
        p1.setStrokeWidth(3);  
          
        p2 = **new** Paint(p1);  
        p2.setColor(Color.GREEN);  
        p2.setPathEffect(pe1);  
          
        p3 = **new** Paint(p1);  
        p3.setColor(Color.BLUE);  
        p3.setPathEffect(pe2);  
          
        p4 = **new** Paint(p1);  
        p4.setColor(Color.RED);  
        p4.setPathEffect(pe3);  
          
        p5 = **new** Paint(p1);  
        p5.setColor(Color.YELLOW);  
        p5.setPathEffect(pe4);  
          
      }  
        
      @Override  
      **protected void** onDraw(Canvas canvas) {  
        canvas.drawARGB(80, 102, 204, 255);  
          
        canvas.translate(120, 100);  
        canvas.drawPath(path, p1);  
          
        canvas.translate(250, 0);  
        canvas.drawPath(path, p2);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p3);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p4);  
          
        canvas.translate(250, 0);  
        canvas.drawPath(path, p5);          
      }  
        
    }

Создаем 4 эффекта.

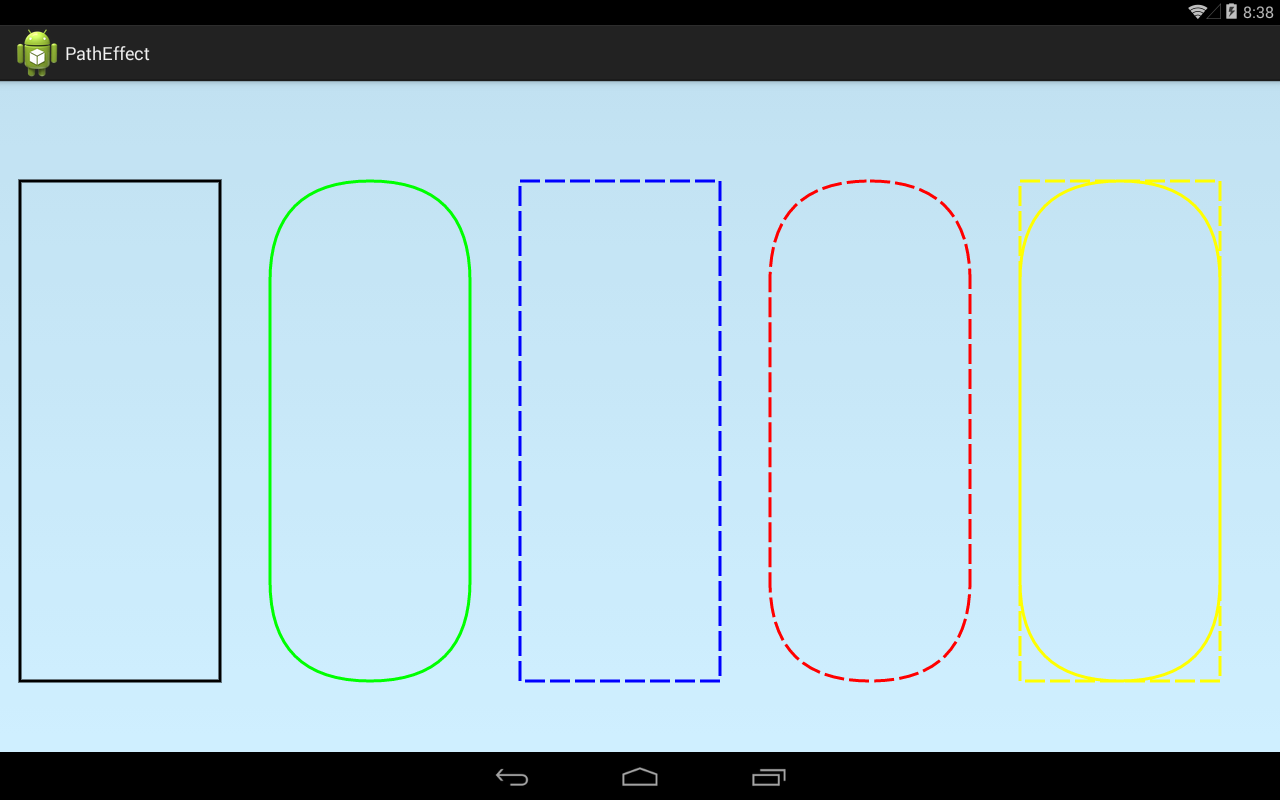
pe1 – закругление

pe2 – прерывистая линяя

pe3 – комбинация, сначала будет применен pe1, затем к получившемуся результату - pe2

pe4 – сумма линия будет нарисована с эффектом pe1 и с эффектом pe2

Результат:



Зеленый прямоугольник закруглен (pe1). Синий нарисован прерывистым (pe2). Красный сначала закруглен, затем сделан прерывистым (pe1, затем pe2). Желтый – просто вывод обоих эффектов отдельно (pe1 и pe2).

В ComposePathEffect имеет значение порядок эффектов. Немного изменим предыдущий пример

**class** DrawView **extends** View {  
        
      Path path;  
      Paint p1;  
      Paint p2;  
      Paint p3;  
      Paint p4;  
      Paint p5;  
        
      **public** DrawView(Context context) {  
        **super**(context);  
        path = **new** Path();  
        path.addRect(-100, 0, 100, 500, Path.Direction.CW);  
          
          
        PathEffect pe1 = **new** CornerPathEffect(100);  
        PathEffect pe2 = **new** DiscretePathEffect(15, 10);  
        PathEffect pe3 = **new** ComposePathEffect(pe1, pe2);  
        PathEffect pe4 = **new** ComposePathEffect(pe2, pe1);  
          
          
        p1 = **new** Paint(Paint.ANTI\_ALIAS\_FLAG);  
        p1.setStyle(Paint.Style.STROKE);  
        p1.setStrokeWidth(7);  
          
        p2 = **new** Paint(p1);  
        p2.setColor(Color.GREEN);  
        p2.setPathEffect(pe1);  
          
        p3 = **new** Paint(p1);  
        p3.setColor(Color.BLUE);  
        p3.setPathEffect(pe2);  
          
        p4 = **new** Paint(p1);  
        p4.setColor(Color.RED);  
        p4.setPathEffect(pe3);  
          
        p5 = **new** Paint(p1);  
        p5.setColor(Color.YELLOW);  
        p5.setPathEffect(pe4);  
          
      }  
        
      @Override  
      **protected void** onDraw(Canvas canvas) {  
        canvas.drawARGB(80, 102, 204, 255);  
          
        canvas.translate(120, 100);  
        canvas.drawPath(path, p1);  
          
        canvas.translate(250, 0);  
        canvas.drawPath(path, p2);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p3);  
  
        canvas.translate(250, 0);  
        canvas.drawPath(path, p4);  
          
        canvas.translate(250, 0);  
        canvas.drawPath(path, p5);          
      }  
        
    }

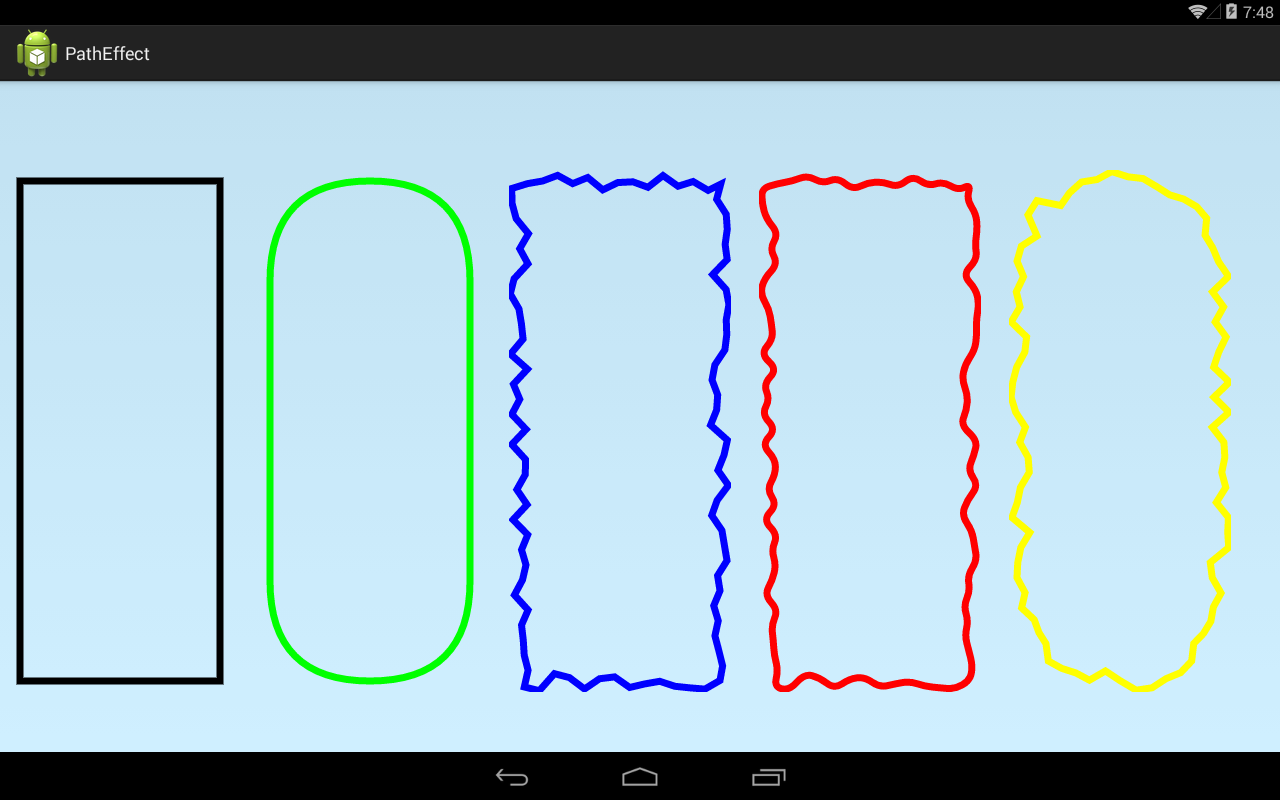
pe1 – закругление

pe2 – излом

pe3 – сначала применен pe2, затем pe1

pe4 – сначала применен pe1, затем pe2

Результат:



Красная фигура – это результат применения сначала излома, затем закругления. Т.е. закругление было применено уже к излому и он получился сглаженным.

А желтая фигура – результат применения сначала закругления, затем излома. Т.е. излом был применен к уже закругленному прямоугольнику.

Я везде использовал Path, но эти эффекты можно применять и при рисовании объектов канвы, например, Canvas.drawRect или Canvas.drawCircle