**[Урок 148. Рисование. Canvas, clip](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/326-urok-148-risovanie-canvas-clip.html)**

В этом уроке:

- используем clip

Обычно нам для рисования доступна вся канва. Но бывают случаи, когда надо нарисовать объект так, чтобы прорисовалась только какая-то его часть, а остальное – нет.

Из жизни можно провести аналогию с двумя бумажными листами. Кладем один лист на стол. А во втором вырезаем отверстие нужной нам формы и кладем второй лист поверх на первый. Далее полностью закрашиваем второй лист краской и убираем его. В итоге получаем на первом листе нужную нам форму ровно закрашенную.

Вот что-то подобное делает clip. Он на канве определяет область, где рисование будет работать. А в остальных местах ничего рисоваться не будет.

Создадим проект:

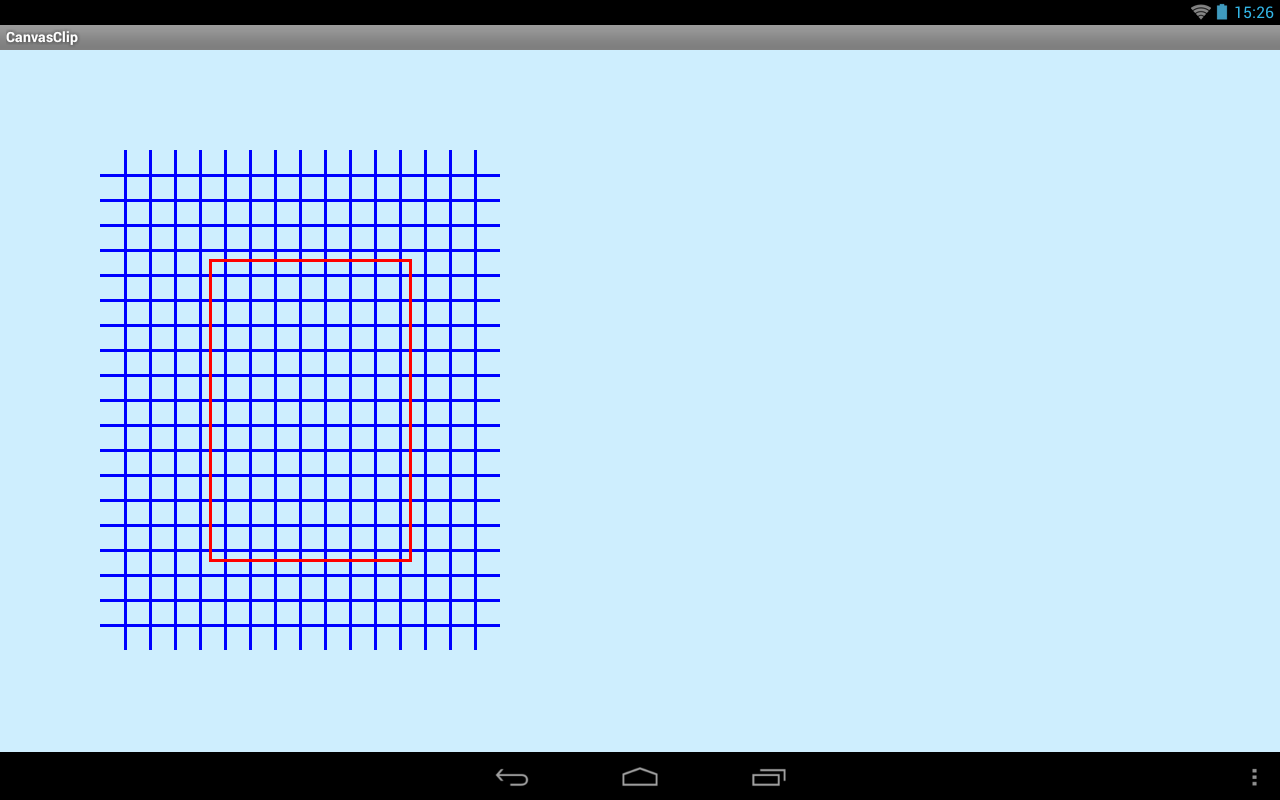
**Project name**: P1481\_CanvasClip  
**Build Target**: Android 2.3.3   
**Application name**: CanvasClip  
**Package name**: ru.startandroid.develop.p1481canvasclip  
**Create Activity**: MainActivity

**MainActivity.java**:

**package** ru.startandroid.develop.p1481canvasclip;  
  
**import** android.app.Activity;  
**import** android.content.Context;  
**import** android.graphics.Canvas;  
**import** android.graphics.Color;  
**import** android.graphics.Paint;  
**import** android.graphics.Rect;  
**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 {  
  
    Paint p;  
    Rect rect;  
  
    **public** DrawView(Context context) {  
      **super**(context);  
      p = **new** Paint();  
      p.setStyle(Paint.Style.STROKE);  
      p.setStrokeWidth(3);  
      rect = **new** Rect(210, 210, 410, 510);  
    }  
  
    @Override  
    **protected void** onDraw(Canvas canvas) {  
      canvas.drawARGB(80, 102, 204, 255);  
  
      // сетка  
      p.setColor(Color.BLUE);  
      drawGrid(canvas);  
  
      // красный прямоугольник  
      p.setColor(Color.RED);  
      canvas.drawRect(rect, p);  
    }  
  
    **private void** drawGrid(Canvas canvas) {  
      **for** (**int** i = 25; i < 400; i += 25) {  
        canvas.drawLine(100 + i, 100, 100 + i, 600, p);  
      }  
      **for** (**int** i = 25; i < 500; i += 25) {  
        canvas.drawLine(100, 100 + i, 500, 100 + i, p);  
      }  
  
    }  
  
  }  
  
}

Просто выводим на экран сетку из линий синим цветом и прямоугольник rect красным цветом.

Результат:



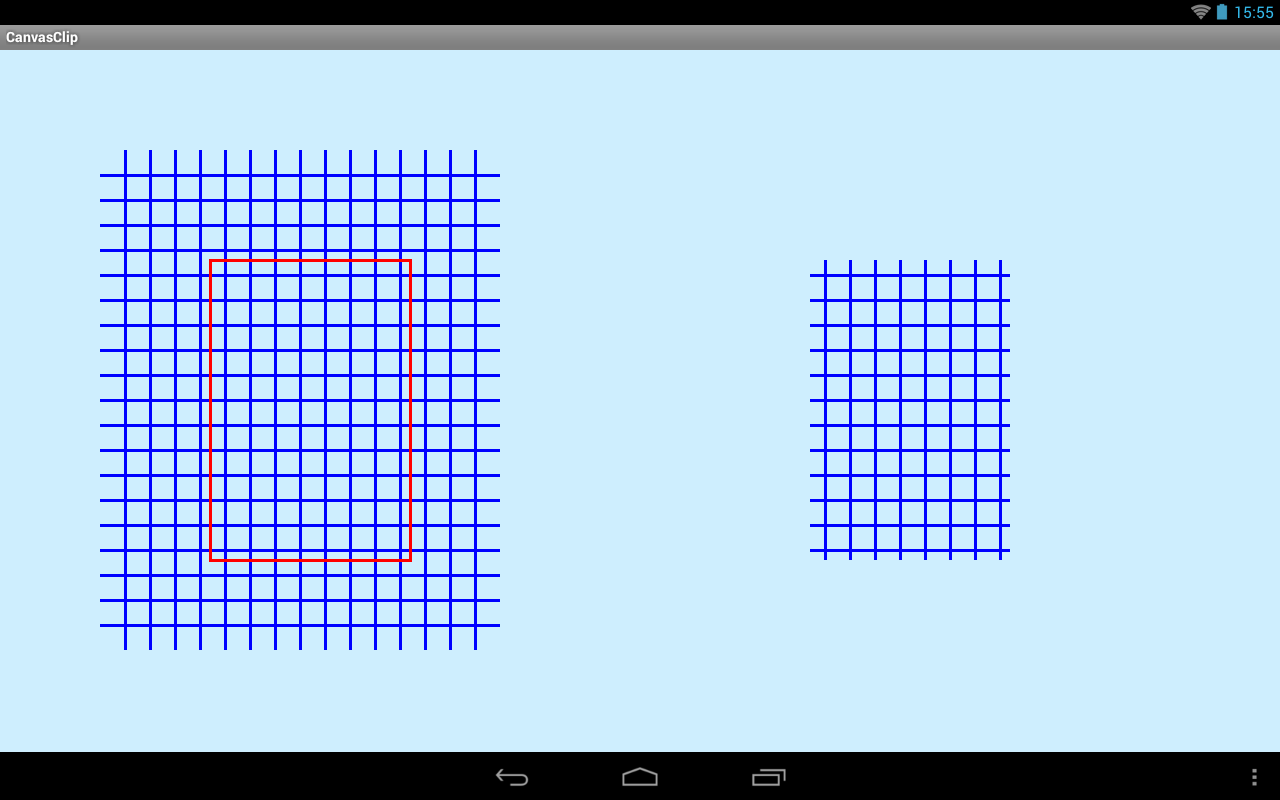
Изменим пример так, чтобы красный прямоугольник ограничивал область рисования на канве.

Перепишем **onDraw**:

    @Override  
    **protected void** onDraw(Canvas canvas) {  
      canvas.drawARGB(80, 102, 204, 255);  
        
      // сетка  
      p.setColor(Color.BLUE);  
      drawGrid(canvas);  
        
      // красный прямоугольник  
      p.setColor(Color.RED);  
      canvas.drawRect(rect, p);  
        
      // смещение  
      canvas.translate(600, 0);  
        
      // задание clip-области  
      canvas.clipRect(rect);  
        
      // сетка   
      p.setColor(Color.BLUE);  
      drawGrid(canvas);  
        
    }

Мы снова выводим синие линии и красный прямоугольник. Затем смещаемся вправо и вместо рисования красного прямоугольника мы методом [clipRect](http://developer.android.com/reference/android/graphics/Canvas.html" \l "clipRect(android.graphics.Rect)" \t "_blank) говорим канве, что теперь рисование доступно только в этой области. И рисуем синие линии.

Результат:



Видим, что справа линии нарисовались только в области rect, т.к. мы задали это методом clipRect.

Разумеется, clip-область будет работать только для последующих рисований. Все, что было нарисовано до задания этой области, остается как было.

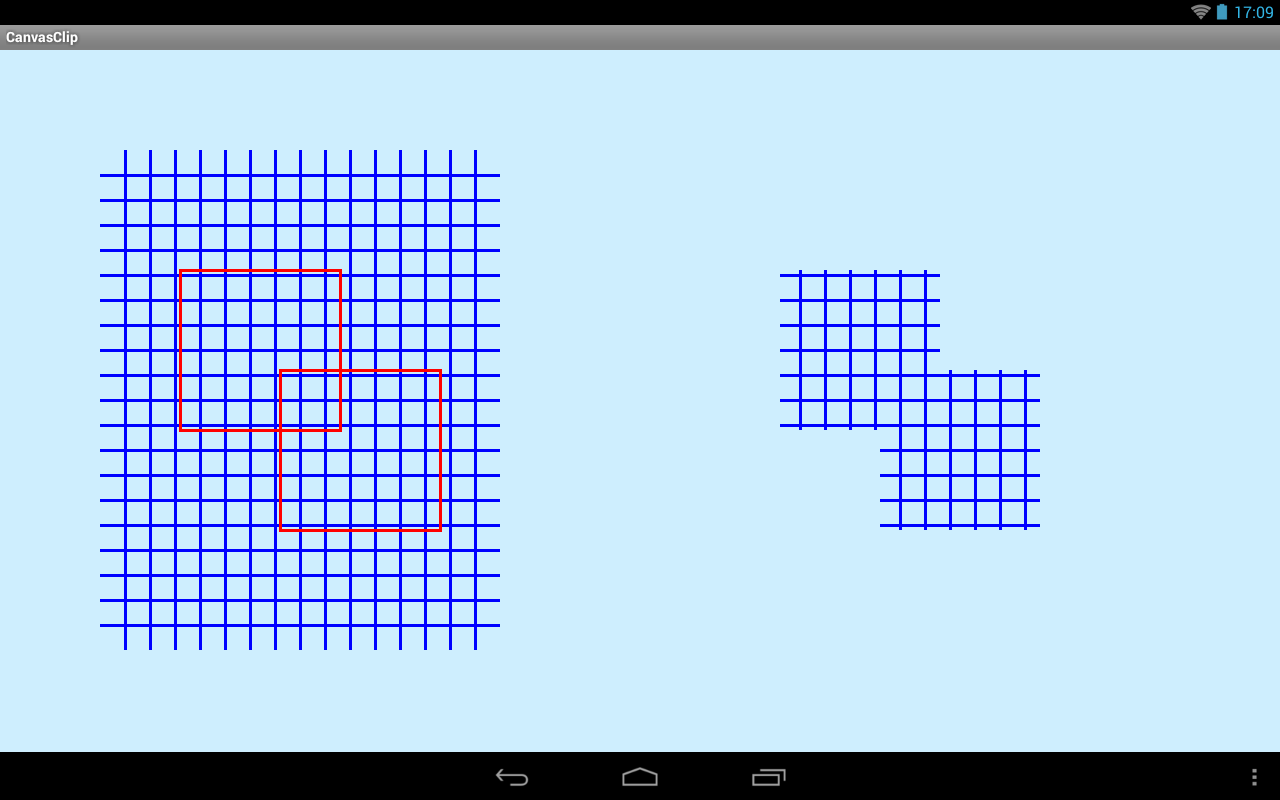
При задании clip-области мы можем использовать несколько прямоугольников. Для добавления прямоугольников используется эта версия метода: [clipRect(Rect rect, Region.Op op)](http://developer.android.com/reference/android/graphics/Canvas.html" \l "clipRect(android.graphics.Rect, android.graphics.Region.Op)" \t "_blank). Здесь используются режимы взаимодействия областей, которые мы подробно рассмотрели в прошлом уроке про регионы.

Посмотрим на примере. Перепишем класс **DrawView**:

**class** DrawView **extends** View {  
  
    Paint p;  
    Rect rect1;  
    Rect rect2;  
    Region.Op op = Region.Op.UNION;  
  
    **public** DrawView(Context context) {  
      **super**(context);  
      p = **new** Paint();  
      p.setStyle(Paint.Style.STROKE);  
      p.setStrokeWidth(3);  
      rect1 = **new** Rect(180, 220, 340, 380);  
      rect2 = **new** Rect(280, 320, 440, 480);  
    }  
  
    @Override  
    **protected void** onDraw(Canvas canvas) {  
      canvas.drawARGB(80, 102, 204, 255);  
  
      // сетка  
      p.setColor(Color.BLUE);  
      drawGrid(canvas);  
  
      // красные прямоугольники  
      p.setColor(Color.RED);  
      canvas.drawRect(rect1, p);  
      canvas.drawRect(rect2, p);  
  
      // смещение  
      canvas.translate(600, 0);  
  
      // задание clip-области  
      canvas.clipRect(rect1);  
      canvas.clipRect(rect2, op);  
  
      // сетка  
      p.setColor(Color.BLUE);  
      drawGrid(canvas);  
  
    }  
  
    **private void** drawGrid(Canvas canvas) {  
      **for** (**int** i = 25; i < 400; i += 25) {  
        canvas.drawLine(100 + i, 100, 100 + i, 600, p);  
      }  
      **for** (**int** i = 25; i < 500; i += 25) {  
        canvas.drawLine(100, 100 + i, 500, 100 + i, p);  
      }  
    }  
  
  }

При использовании UNION, области прямоугольников объединились в одну область и она стала clip-областью.

Результат:



Чтобы убрать clip-область, можно использовать метод [restore](http://developer.android.com/reference/android/graphics/Canvas.html" \l "restore()" \t "_blank). Все те механизмы сохранения/восстановления канвы, что мы рассмотрели в Уроке 146 относятся не только к матрице, но и к clip.

Я в этом уроке использовал объекты Rect. Но у канвы существуют и другие версии метода задания clip-области. Можно использовать RectF, Region и Path.