Android程序设计

画布

2019.6.21

isszym　sysu.edu.cn

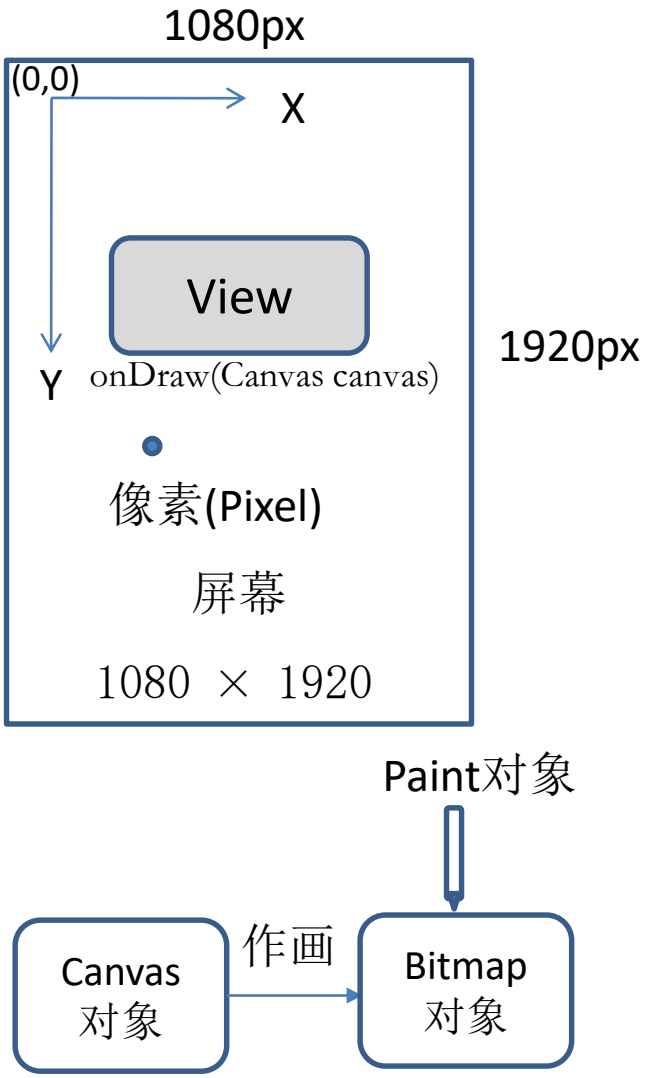
官方文档（中文） 官方文档（英文） runoob cnblogs

Canvas画图

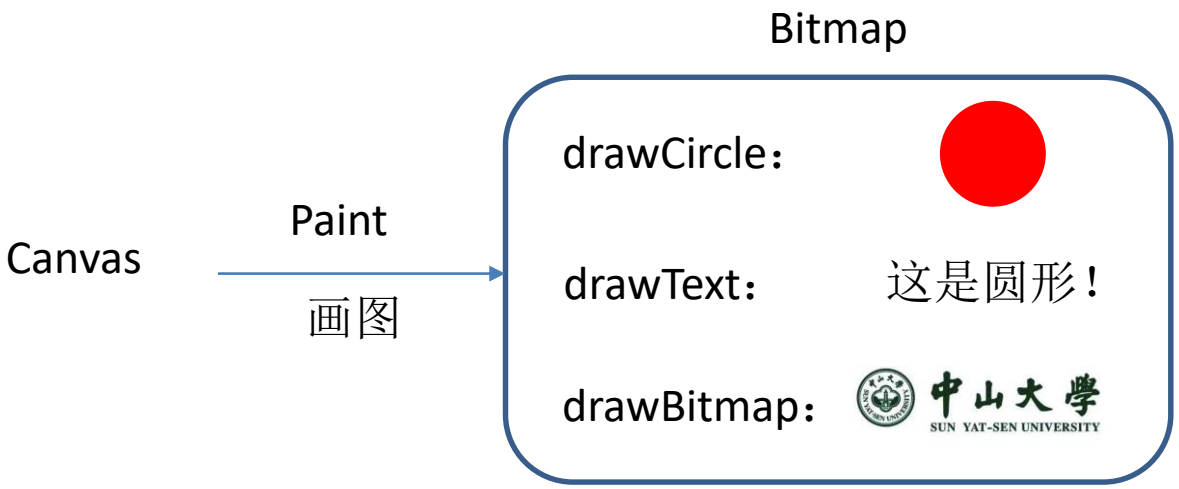
参考

视图、画布和位图

- 每个视图（View）对象默认都有一个画布（**Canvas**）对象，并且关联一个位图（**Bitmap**）对象。利用画布对象可以在其关联的位图对象上用画笔（**Paint**）作画。
- 使用控件（例如，Button）时系统会预先在控件的位图对象上作画（画一个按钮），系统还给出回调函数onDraw()，让用户可以利用其画布在控件上作图。
- 用户还可以创建Canvas对象并关联一个Bitmap对象，然后对该Bitmap对象进行作画。最后通过回调函数onDraw()把这个Bitmap对象画到一个View上就可以显示出来。
- Bitmap对象的每个像素默认采用4个字节表示(ARGB)，即Bitmap.Config. ARGB_8888格式。其他格式很少用，具体见附录。



- 利用Canvas的方法可以在Bitmap上画形状，例如，画圆(drawCircle)，还可以画文字和画图像。



- 如果希望多次画出一批形状，可以先把多个形状组成**路径(path)**再一起画出来。
- 在画图像前可以对位图进行平移(translate)、旋转（rotate）、缩放(scale)、倾斜(Skew)、过滤(filter)、叠加(overlap)等操作（**Matrix变换**）。
- 如果希望把一批图像进行相同的平移操作，可以把画布先进行平移（**Canvas变换**），再绘制这批图像，最后恢复画布为原来的状态。进行其他操作也一样。
- 通过**颜色过滤器**可以转换绘制图像的颜色，绘制到目标图像的**叠加方式(PorterDuff)**也可以进行多种定义。

画图的例子

- 这个例子定义了两个View的子类MyView和MyView2以及一个Button的子类MyButton。它们都是通过View的onDraw()回调函数中View的Canvas对象作画。
- MyView画一个圆、一条线、一段文字和一个图片。
- MyView2通过触摸事件在自定义的位图上作画，然后用调用invalidate()激活onDraw()，最后利用在onDraw()中把自定义位图画到到视图位图上显示出来。
- MyButton在onDraw()中绘制了一个矩形。

MyView

MyView2

MyButton

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public void getPermission(){  // 动态获取权限，Android 6.0 新特性，一些保护
        //权限，除了要在AndroidManifest中声明权限，  还要使用如下代码动态获取
        if (Build.VERSION.SDK_INT >= 23) {
            int REQUEST_CODE_CONTACT = 101;
            String[] permissions={Manifest.permission.WRITE_EXTERNAL_STORAGE};
            for (String str : permissions) {          //验证是否许可权限
                if(this.checkSelfPermission(str)
                    !=PackageManager.PERMISSION_GRANTED){//申请权限
                    this.requestPermissions(permissions, REQUEST_CODE_CONTACT);
                    return;
                }
            }
        }
    }
    protected void saveBitmap(View vw) {
        MyView2 view = (MyView2) findViewById(R.id.my_view2);
        SaveImage(view.getBitmap());
        return;
    }
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"  >
    <com.example.isszym.drawalloncanvas.MyView
        android:id="@+id/my_view"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:background="@android:color/background_light"
        />
    <com.example.isszym.drawalloncanvas.MyView2
        android:id="@+id/my_view2"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:background="@android:color/darker_gray"/>
    <com.example.isszym.drawalloncanvas.MyButton
        android:id="@+id/btn_save"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:textSize="16dp"
        android:text="Save"
        android:onClick="saveBitmap"
        />
</LinearLayout>
```

```
public class MyView extends View {
    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();                  // 创建新画笔
        paint.setColor(Color.BLUE);                 // 设置画笔颜色
        canvas.drawCircle(180, 200, 160, paint);  // 画圆

        paint.setColor(Color.GREEN);

        paint.setStrokeWidth(8);                    // 设置画笔粗细
        canvas.drawLine(100,154,400,400,paint);    // 画线

        paint.setColor(Color.RED);
        paint.setTextSize(100);                     //设置文字大小
        paint.setStyle(Paint.Style.STROKE);       // 设置画笔样式(空心）.默认为FILL
        canvas.drawText("Hello, world!",360,340,paint); //画出一段文字

        Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(),R.raw.sysu);
        canvas.drawBitmap(bitmap,820,340,new Paint());  //  贴图到画布上
    }
}
```

MyView2.java

```
public class MyView2 extends View {
    final String TAG ="画图";
    private Paint mPaint = null;
    private Bitmap mBitmap = null;
    private Canvas mBitmapCanvas = null;
    private float startX;
    private float startY ;
    public MyView2(Context context, AttributeSet attrs) {
       super(context, attrs);
       mBitmap = Bitmap.createBitmap(1200,800, Bitmap.Config.ARGB_8888);
       mBitmapCanvas = new Canvas(mBitmap);
       mBitmapCanvas.drawColor(Color.GRAY);
       mPaint = new Paint();
       mPaint.setColor(Color.RED);
       mPaint.setStrokeWidth(6);
    }

    @Override
    protected void onDraw(Canvas canvas) {
       if(mBitmap != null) {
          canvas.drawBitmap(mBitmap, 0, 0, mPaint);
       }
    }
```

```
    public Bitmap getBitmap(OutputStream stream) {
        return mBitmap;
     }
     @Override
     public boolean onTouchEvent(MotionEvent event) {
       switch (event.getAction()) {
         case MotionEvent.ACTION_DOWN:
            startX = event.getX();
            startY = event.getY();
            break;
         case MotionEvent.ACTION_MOVE:
            float stopX = event.getX();
            float stopY = event.getY();
            Log.e(TAG,"onTouchEvent-ACTION_MOVE\nstartX is "+startX
              +" startY is "+startY+" stopX is "+stopX+ " stopY is "+stopY);
            mBitmapCanvas.drawLine(startX, startY, stopX, stopY, mPaint);
            startX = event.getX();
            startY = event.getY();
            invalidate();     //call onDraw()
            break;
       }
       return true;
     }
}
```

```
    private void SaveImage(Bitmap finalBitmap) {
        File file=getFile();
        try {
             FileOutputStream out = new FileOutputStream(file);
             finalBitmap.compress(Bitmap.CompressFormat.JPEG, 90, out);
             out.flush(); out.close();
        } catch (Exception e) {
             Toast.makeText(this, "save failed! "+e.getMessage(),
                            Toast.LENGTH_SHORT).show();
             e.printStackTrace();
        }
    }
    private File getFile(){
         getPermission();
         String root = Environment.getExternalStorageDirectory().toString();
         File myDir = new File(root + "/Images");
         if(!myDir.mkdirs()) Toast.makeText(this, "mkdirs failed! ",
                            Toast.LENGTH_SHORT).show();
         Random generator = new Random();
         int n = 10000;  n = generator.nextInt(n);
         String fname = "Image-"+ n +".jpg";
         File file = new File (myDir, fname);//if(file.exists()) file.delete();
         return file;
    }
}
```

```
public class MyButton extends Button {
    public MyButton(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();

        paint.setColor(Color.argb(200,200,80,80));
        paint.setStrokeWidth(4);
        paint.setStyle(Paint.Style.STROKE);
        canvas.drawRect(new Rect(40, 40, 1040, 260) , paint);
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.isszym.newcanvas">
    <uses-permission android:name="android.permission.MOUNT_FORMAT_FILESYSTEMS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    </application>
</manifest>
```

画笔Paint

参考

- 创建画笔

```
Paint paint = new Paint();
```

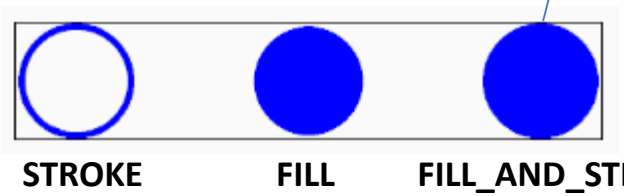
- 设置画笔：
  - 可以为画笔设置颜色（setARGB和setColor）、透明度（setAlpha）、笔画的粗细（setStrokeWidth ）和**样式（setStyle）。通过设置画笔样式可以画出**实心图(FILL)、轮廓图（STROKE）和实心且带轮廓的图（FILL_AND_STROKE）。
  - 可以用画笔做各种渐变图（setShader）、加上阴影（setShadowLayer）。
  - 可以用画笔变换颜色（setColorFilter），进行模糊变换（ setMaskFilter ）。

- 画笔的常用属性
  - 设置画笔的透明度和颜色
    Paint paint = new Paint();
    setARGB(int a, int r, int g, int b)
    例如，paint.setARGB(128,255,255,0);
  - 设置画笔的透明度
    setAlpha(int a)
    例如，paint.setAlpha(80);
  - 设置画笔的颜色
    setColor(int color)
    例如：paint.setColor(Color.*BLUE*);
  - 设置画笔粗细
    setStrokeWidth(int width)
    例如，paint.setStrokeWidth(10); //像素数
  - 设置画笔的样式
    setStyle(Paint.Style style)
    例如：paint.setStyle(Paint.Style.***STROKE***);
  - 设置是否抗锯齿
    setAntiAlias(Boolean aa);
  - 设置是否过滤图像(抗锯齿)
    setFilterBitmap(Boolean fb)

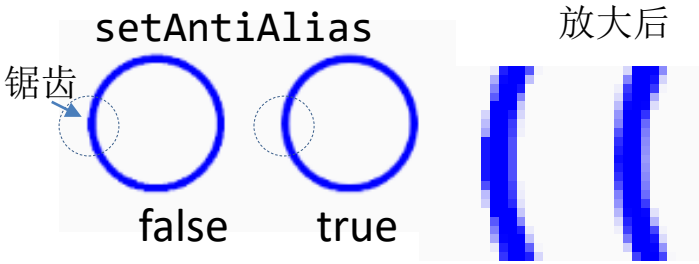
a、r、g、b分别代表不透明度(alpha)、红色(Red)、绿色(Green)、蓝色(Blue)，取值均在 0~255 之间。

a(不透明度)的取值:0~255。

比FILL多了一个同色边界




```
paint.setColor(Color.BLUE);
paint.setStrokeWidth(10);
```



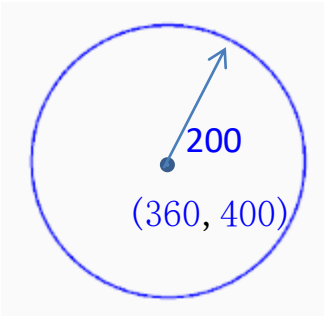
绘制形状

- 概述

  通过定义View或者控件的子类，可以利用其继承的回调函数onDraw(Canvas canvas)获得该View的画布(Canvas)，然后用生成的画笔(Paint)就可以在该画布上画点、划线、画圆、画矩形等，既可以画轮廓图也可以画成实心图。

- 在Canvas上绘制圆形

```
public class ShapeView extends View {
    public ShapeView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();
        paint.setStrokeWidth(4);              // 画笔粗细
        paint.setColor(Color.BLUE);           // 画笔颜色
        paint.setStyle(Paint.Style.STROKE); // 轮廓样式
        canvas.drawCircle(360, 400, 200, paint);
    }
}
```



参考

参考

- 绘制形状的语句

RectF为浮点矩形，采用四个点代替它要求版本 >= level 21，在低版本四个点要加上@SuppressLint("NewApi")防止出错。

（1）画点

drawPoint(float x, float y, Paint paint);

（2）画线

canvas.drawLine(float startX, float startY, float stopX, float stopY, Paint paint);

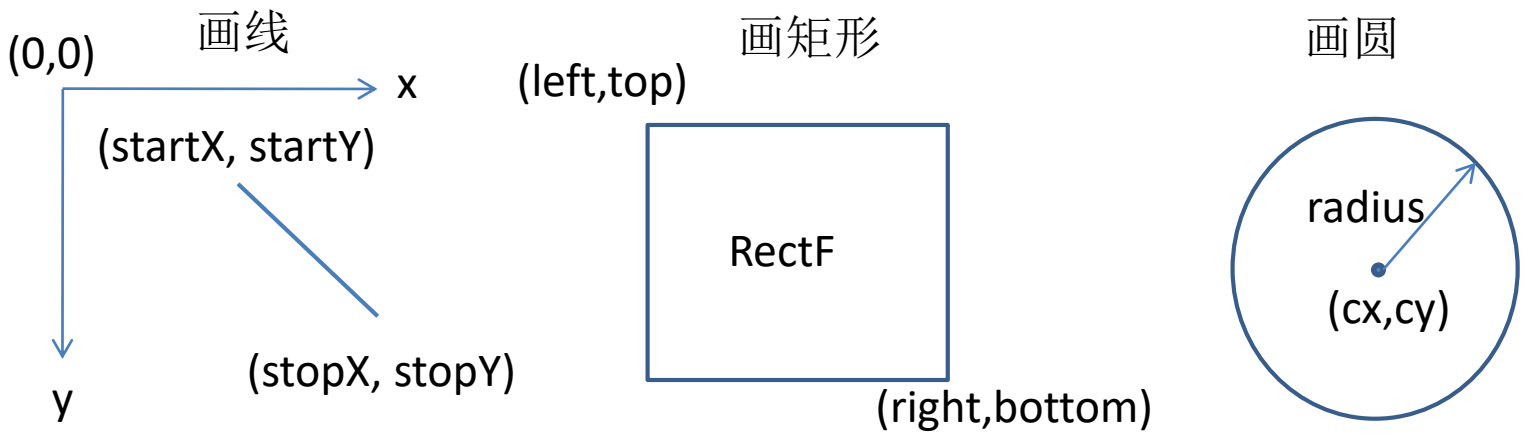
（3）画矩形

canvas.drawRect(RectF rect, Paint paint);

canvas.drawRect(float left, float top, float right, float bottom, Paint paint);

（4）画圆

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



参考

（5）画椭圆

drawOval(RectF oval, Paint paint);

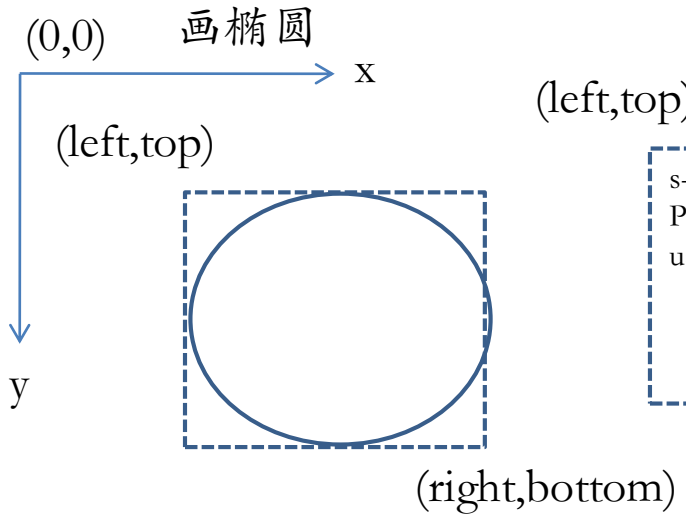
drawOval(float left, float top, float right, float bottom, Paint paint);

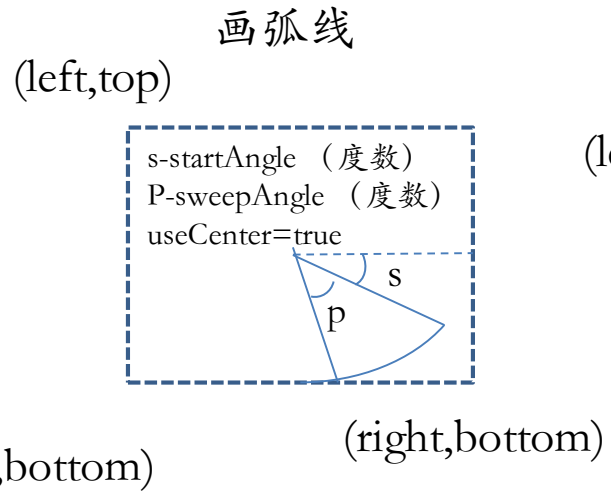
（6） 画弧

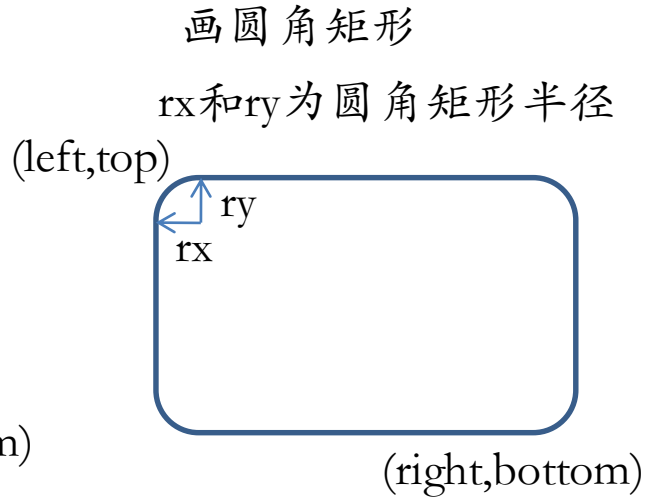
drawArc(RectF oval, float startAngle, float sweepAngle,boolean useCenter, Paint paint)

（7）画圆角矩形

drawRoundRect(RectF rect, float rx, float ry, Paint paint) // rx,ry 圆角半径







（8）一次画多个点

drawPoints(float[] pts, Paint paint); // new float[]{x0,y0,x1,y1,….}

drawPoints(float[] pts, int offset, int count,Paint paint); //从offset开始画count个

（9）一次画多条线

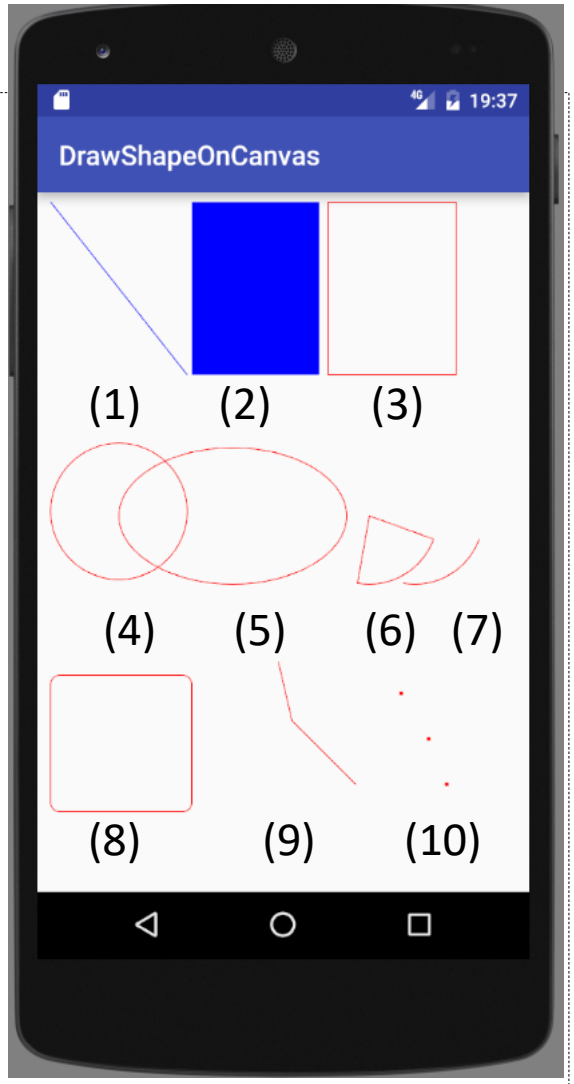
drawLines(float[] pts, Paint paint); // new float[]{x0,y0,x1,y1,….}

drawLines(float[] pts, int offset, int count,Paint paint);

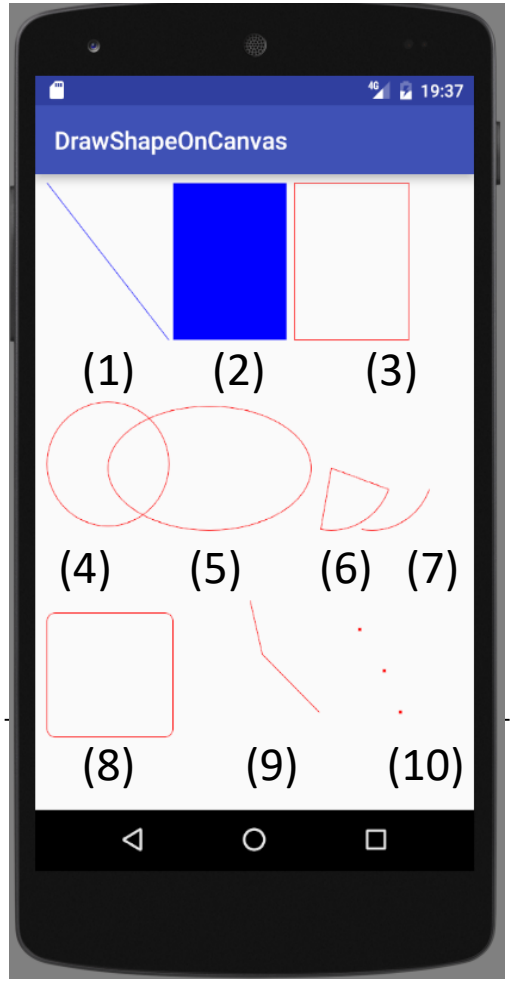
- 形状绘制的例子

ShapeView.java

```
public class ShapeView extends View {
    public ShapeView(Context context,
                     AttributeSet attrs) {
        super(context, attrs);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();    // 创建新画笔
        paint.setColor(Color.BLUE);   // 设置画笔颜色
        paint.setStrokeWidth(2);      // 设置画笔粗细
        paint.setAntiAlias(true);     // 设置抗锯齿
         //(1)画线：左上角(left,top)右下角(right,bottom)
        canvas.drawLine(10, 20, 300, 400, paint);
        //(2)矩形：左上角(320,20)，右下角(600,400)
        canvas.drawRect(320, 20, 600, 400,paint);

       // 设置画笔样式(空心) .默认为FILL
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.RED); // 设置画笔颜色
        canvas.drawRect(new RectF(620f, 20f, 900f, 400f), paint); //(3)画矩形
        canvas.drawCircle(160, 700, 150, paint); //(4)圆心(160,700)，半径150
```



```
        //(5)左上角和右下角.
        canvas.drawOval(new RectF(160, 560, 660, 860), paint);
        //(6)画弧线：椭圆,起始度数,顺时钟扫过的度数,用半径线进行封闭
        canvas.drawArc(new RectF(560f,560f,860f,860f),20f,80f,true,paint);
        //(7)画弧线：椭圆,起始度数,顺时钟扫过的度数,不用半径线进行封闭
        canvas.drawArc(new RectF(660f,560f,960f,860f),
                       20f, 80f, false,paint);
        //(8)圆角矩形:Rect，圆角x半径，圆角y半径
        canvas.drawRoundRect(new RectF(10f, 1060f, 320f,
                       1360f),20f, 20f, paint);
        //(9)一次画多条线，每四个数一条线
        canvas.drawLines(new float[]{530,1030,560,1160,
                      560,1160,700,1300}, paint);
        //(10)一次画多个点
        paint.setStrokeWidth(8);
        canvas.drawPoints(new float[]{800,1100,
                       860,1200,900,1300}, paint);
    }
}
```



actitvity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.example.isszym.drawshapeoncanvas.ShapeView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

MainActitvity.java

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
    }
}
```

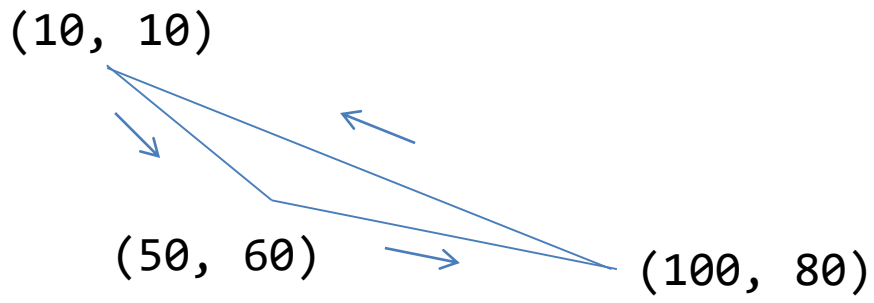
绘制路径

Path参考1 参考2 参考3 参考4

- 概述

与每次绘制一个特定图形不同，路径可以把一次形成多个形状，然后把它绘制出来。一次形成的路径可以在多个地方重复绘制出来。

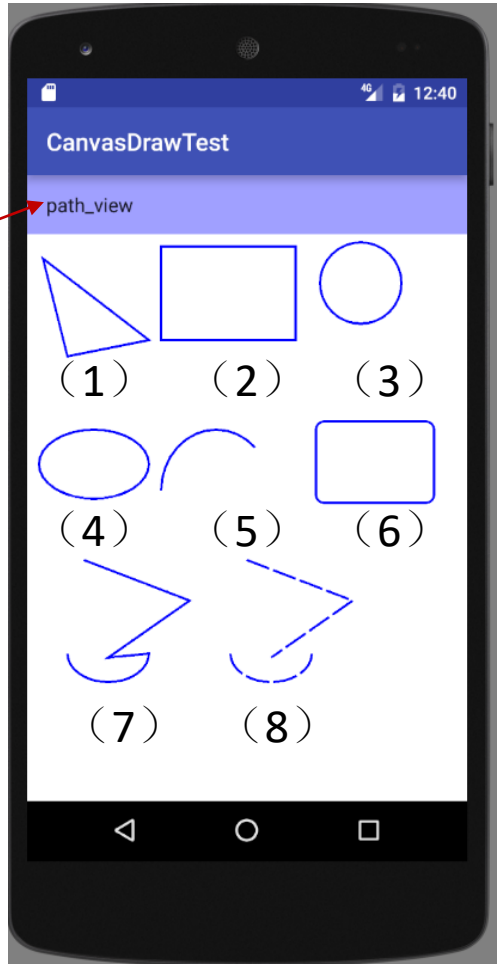
```
Path path = new Path(); //定义一条路径
path.moveTo(10, 10);    // 移动到坐标10,10
path.lineTo(50, 60);    // 再到坐标50,60
path.lineTo(100,80);    // 再到坐标100,80
path.lineTo(10, 10);
canvas.drawPath(path, paint);
```



- 绘制路径的例子(PathView)

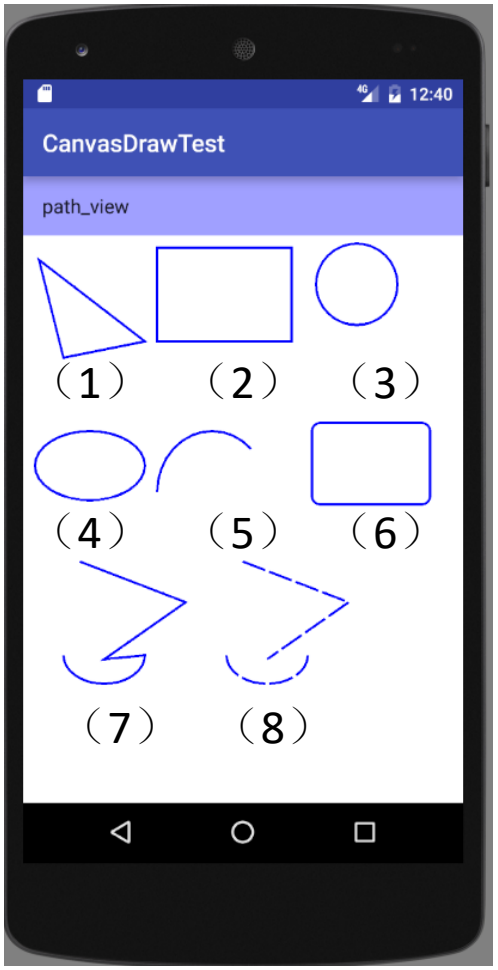
```
Paint paint = new Paint();
paint.setColor(Color.BLUE);
paint.setStrokeWidth(6);
Path path = new Path();
// (1)
paint.setStyle(Paint.Style.FILL);
path.moveTo(40, 60);        // 移动到坐标340,1060
path.lineTo(300, 260);      // 再到坐标900,1260
path.lineTo(100, 300);      // 再到坐标800,1400
path.close();
// (2)画矩形：左上角，右下角
paint.setStyle(Paint.Style.STROKE);
path.addRect(330, 30, 660, 260, Path.Direction.CCW);
// (3) 画圆：圆心坐标，半径
path.addCircle(820, 120, 100, Path.Direction.CCW);
// (4)画椭圆：左上角，右下角
path.addOval(30, 480, 300, 650, Path.Direction.CW);
 //(5) 画弧线：左上角，右下角，起始角度，扫描角度
path.addArc(330, 480, 600, 780, 180, 135);//L T R B startAngle sweepAngle
 //(6)画圆角矩形：左上角，右下角，x方向圆角，y方向圆角
path.addRoundRect(710,460,1000,660,20,20,Path.Direction.CW);//LTRB rx ry
canvas.drawPath(path, paint); // 把上面加入的路径在画布上画出来
```

* 后面很多例子都是在项目CanvasDrawTest中，需选择



* CCW--counter-clockwise CW-- clockwise

```
//(7)
path.reset();
canvas.translate(0,800);
path.moveTo(140, 0);
path.lineTo(400, 100);
path.lineTo(200, 240);
path.arcTo(100, 160, 300, 300, 0, 180, false);
canvas.drawPath(path, paint);

//(8)
path.reset();
canvas.translate(400, 0);
paint.setPathEffect(new DashPathEffect(
          new float[]{60, 10}, 1));//虚线长，空隙长
path.moveTo(140, 0);
path.lineTo(400, 100);
path.lineTo(200, 240);
path.arcTo(100, 160, 300, 300, 0, 180, true);
canvas.drawPath(path, paint);
```

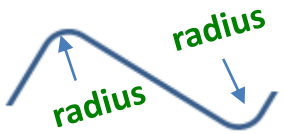


* 由于上面的一些没用RectF定义矩形区域的语句只有高版本有效，需要增加指令@SuppressLint("NewApi")

• 路径效果(PathEffectView)

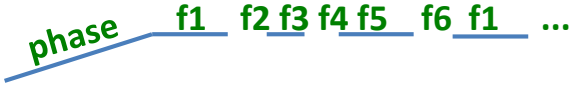
参考1 参考2

利用路径效果，可以画出不同效果的路径。



CornerPathEffect(float radius)

将Path的各个连接线段之间的夹角用平滑方式连接。Radius指连接的圆弧半径。



DashPathEffect(float[] intervals, float phase)

将Path的线段虚线化，intervals为虚线的ON(线段长度)和OFF数组(空隙长度)，可以定义多线段， 比如，形成点线模式，phase（偏移量）是指从偏移指定长度的位置开始画，但总长度还是不变。

DiscretePathEffect(float segmentLength,float deviation)

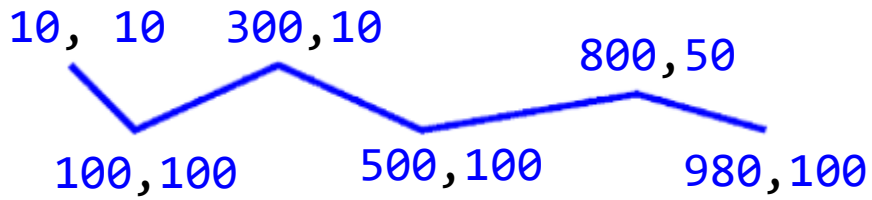
一种切断线段随机偏离模式。segmentLength是指定切断的长度，deviation为切断之后线段的偏移量，随机的，小于等于deviation。

PathDashPathEffect(Path shape, float advance, float phase, PathDashPathEffect.Stylestyle)

shape则是指填充图形，advance指每个图形间的间距，phase为绘制时的偏移量。style为该类自由的枚举值：Style.ROTATE、Style.MORPH和Style.TRANSLATE。其中ROTATE的情况下，线段连接处的图形转换以旋转到与下一段移动方向相一致的角度进行转转，MORPH时图形会以发生拉伸或压缩等变形的情况与下一段相连接，TRANSLATE时，图形会以位置平移的方式与下一段相连接。

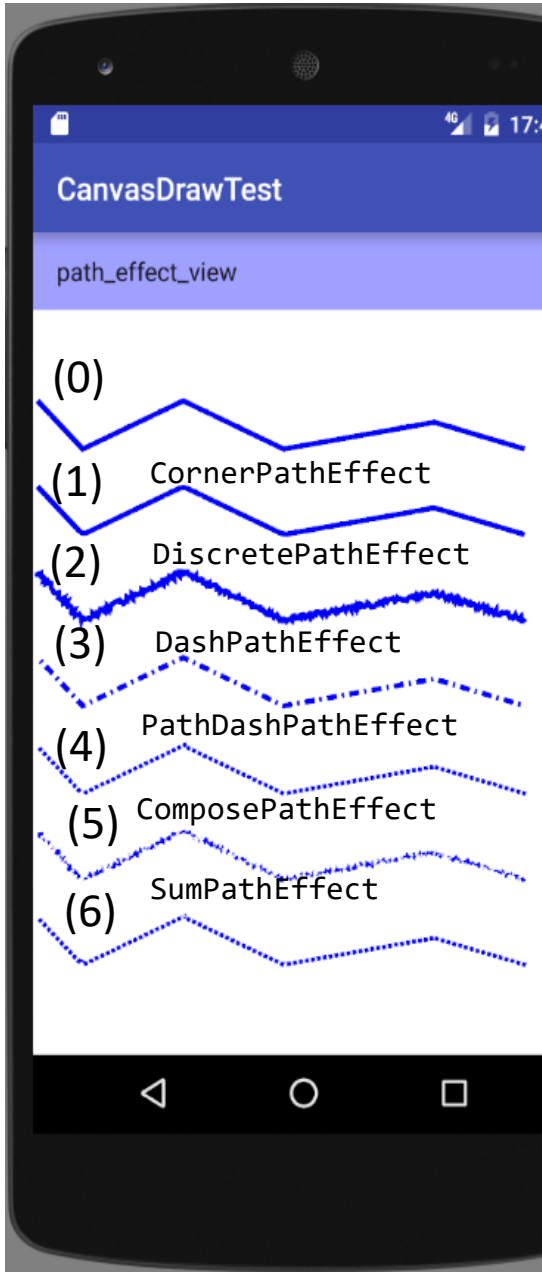
ComposePathEffect和SumPathEffect可以组合或叠加多个路径效果。

例子：把下面曲线用七种效果画出来。




```
public class PathEffectView extends View {
    private float mPhase;// 偏移值
    Path mPath;
    Paint mPaint;
    PathEffect[] mEffects;
    public PathEffectView(Context context,
                          AttributeSet attrs){
        super(context, attrs);
        mPath = new Path();
        mPath.moveTo(10, 10);   mPath.lineTo(100,100);
        mPath.lineTo(300,10);   mPath.lineTo(500,100);
        mPath.lineTo(800,50);   mPath.lineTo(980,100);
        mEffects = new PathEffect[7];
        mPaint = new Paint();
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setStrokeWidth(10);
        mPaint.setColor(Color.BLUE);
    }
```

```
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        mEffects[0] = null;
        mEffects[1] = new CornerPathEffect(10);
        mEffects[2] = new DiscretePathEffect(3.0F,5.0F);
        mEffects[3] = new DashPathEffect(new float[]
                            { 20, 10, 5, 10 }, mPhase);
        Path path1=new Path();
        path1.addRect(0, 0, 8, 8, Path.Direction.CCW);
        mEffects[4]=new PathDashPathEffect(path1,
           12,mPhase,PathDashPathEffect.Style.ROTATE);
        mEffects[5] = new ComposePathEffect(
                            mEffects[2], mEffects[4]);
        mEffects[6] = new SumPathEffect(
                            mEffects[3],mEffects[4]);
        for (int i = 0; i < mEffects.length; i++) {
            canvas.translate(0,160);
            mPaint.setPathEffect(mEffects[i]);
            canvas.drawPath(mPath, mPaint);
        }
        mPhase += 1;
        invalidate(); //要求系统重画
    }
}
```



- 线段头部和接头的形状

setStrokeJoin(Paint.Join join)

设置画笔拐弯处的连接风格，MITER:结合处为锐角，ROUND:结合处为圆弧，BEVEL：结合处为直线。

Paint.Join.***MITER***

Paint.Cap.***BEVEL***

Paint.Cap.***ROUND***

setStrokeCap(cap)

当画笔样式为STROKE或FILL_OR_STROKE时，设置笔刷的图形样式，如圆形样式 Cap.ROUND或方形样式Cap.SQUARE。

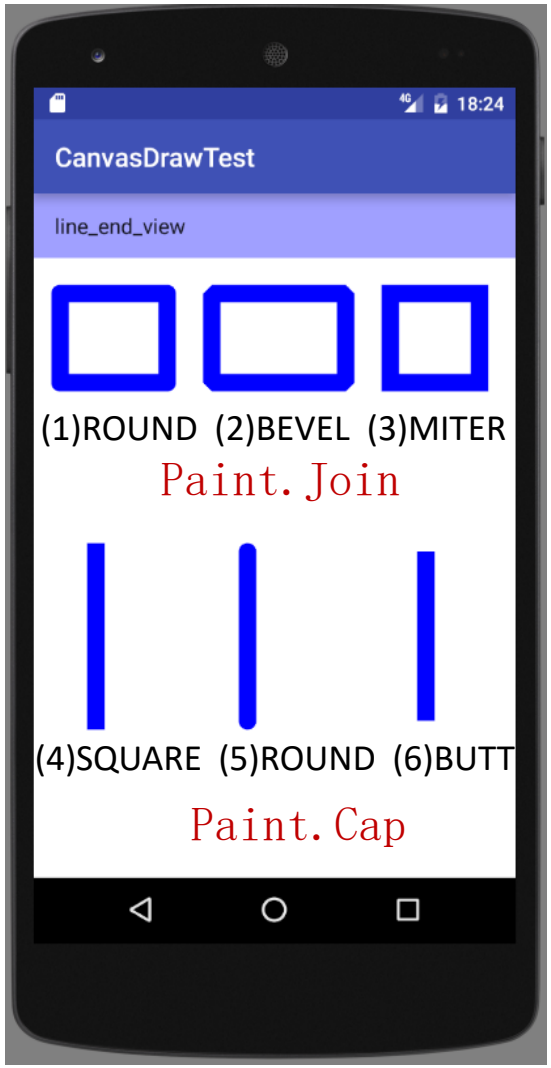
Paint.Cap.***ROUND***

Paint.Cap.***SQUARE***

Paint.Cap.***BUTT***

例子：

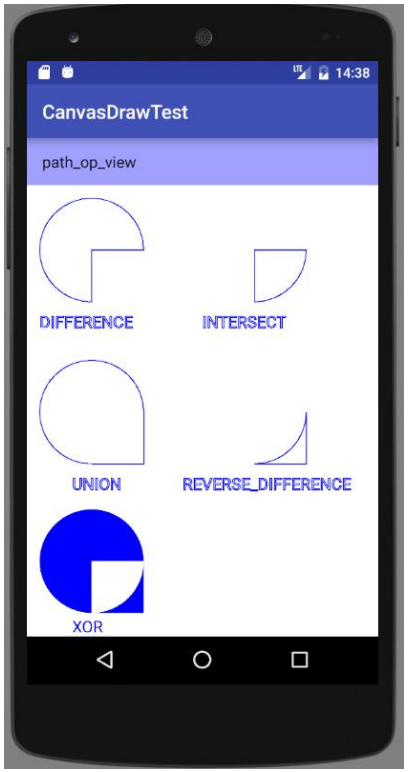
```
public class LineEndView extends View {
    private Paint mPaint;
    public LineEndView(Context context,AttributeSet attrs){super(context, attrs);}
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);  mPaint = new Paint();
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setStrokeWidth(40);
        mPaint.setColor(Color.BLUE);
        mPaint.setStrokeJoin(Paint.Join.ROUND);
        canvas.drawRect(60,80,300,280,mPaint);
        mPaint.setStrokeJoin(Paint.Join.BEVEL);
        canvas.drawRect(400,80,700,280,mPaint);
        mPaint.setStrokeJoin(Paint.Join.MITER);
        canvas.drawRect(800,80,1000,280,mPaint);
        canvas.translate(80,600);
        mPaint.setStrokeCap(Paint.Cap.SQUARE);
        canvas.drawLine(60,60,60,440,mPaint);
        mPaint.setStrokeCap(Paint.Cap.ROUND);
        canvas.drawLine(400,60,400,440,mPaint);
        mPaint.setStrokeCap(Paint.Cap.BUTT);
        canvas.drawLine(800,60,800,440,mPaint);
    }
}
```



- 多路径叠加(Path.Op)

参考

```
public class PathOpView extends View {
    Context context;
    public PathOpView(Context context, AttributeSet attrs){
        super(context, attrs);  this.context = context;
    }
    @SuppressLint("NewApi")
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();
        paint.setColor(Color.BLUE);
        paint.setStrokeWidth(3);
        paint.setStyle(Paint.Style.STROKE);
        paint.setTextSize(50);
        //(1)
        Path path1 = new Path();
        Path  path2  =  new  Path();
        path1.addCircle(200,200,160,Path.Direction.CCW);
        path2.addRect(200, 200, 360,360,Path.Direction.CCW);
        path1.op(path2,  Path.Op.DIFFERENCE);
        canvas.drawPath(path1,  paint);
        canvas.drawText("DIFFERENCE",40,440,paint);
```



* XOR不包含INTERSECT部分，由于取的是轮廓，所以轮廓全部显示出来了

```
        path1.reset(); path2.reset(); //(2)
        path1.addCircle(700, 200, 160, Path.Direction.CCW);
        path2.addRect(700, 200, 860, 360, Path.Direction.CCW);
        path1.op(path2, Path.Op.INTERSECT);
        canvas.drawPath(path1, paint);
        canvas.drawText("INTERSECT",540,440,paint);

        path1.reset(); path2.reset(); //(3)
        path1.addCircle(200, 700, 160, Path.Direction.CCW);
        path2.addRect(200, 700, 360, 860, Path.Direction.CCW);
        path1.op(path2, Path.Op.UNION);
        canvas.drawPath(path1, paint);
        canvas.drawText("UNION",140,940,paint);

        path1.reset(); path2.reset(); //(4)
        path1.addCircle(700, 700, 160, Path.Direction.CCW);
        path2.addRect(700, 700, 860, 860, Path.Direction.CCW);
        path1.op(path2, Path.Op.REVERSE_DIFFERENCE);
        canvas.drawPath(path1, paint);
        canvas.drawText("REVERSE_DIFFERENCE",480,940,paint);

        path1.reset(); path2.reset(); paint.setStyle(Paint.Style.FILL); //(5)
        path1.addCircle(200, 1160, 160, Path.Direction.CCW);
        path2.addRect(200, 1160, 360, 1320, Path.Direction.CCW);
        path1.op(path2, Path.Op.XOR);
        canvas.drawPath(path1, paint);
        canvas.drawText("XOR",140,1380,paint);
    }
}
```

绘制文字

参考

（1）指定文本左上角位置

```
public void drawText (String  text, float x, float y, Paint paint)
public void drawText (String  text, int start, int end, float x, float y, Paint paint)  // 绘制子串
public void drawText (CharSequence  text, int start, int end, float x, float y, Paint paint) // 绘子串
public void drawText (char[] text, int index, int count, float x, float y, Paint paint)
```

```
String str = "ABCDEFGHIJK";
canvas.drawText(str,20,100,paint);       // 文本 x轴坐标 y轴坐标
canvas.drawText(str,1,3,20,200,paint); //字符串 开始位置 结束位置 x y
```

（2）指定每个文字的位置（已过时）

```
public void drawPosText (String text, float[] pos, Paint paint)
public void drawPosText (char[] text, int index, int count, float[] pos, Paint paint)
```

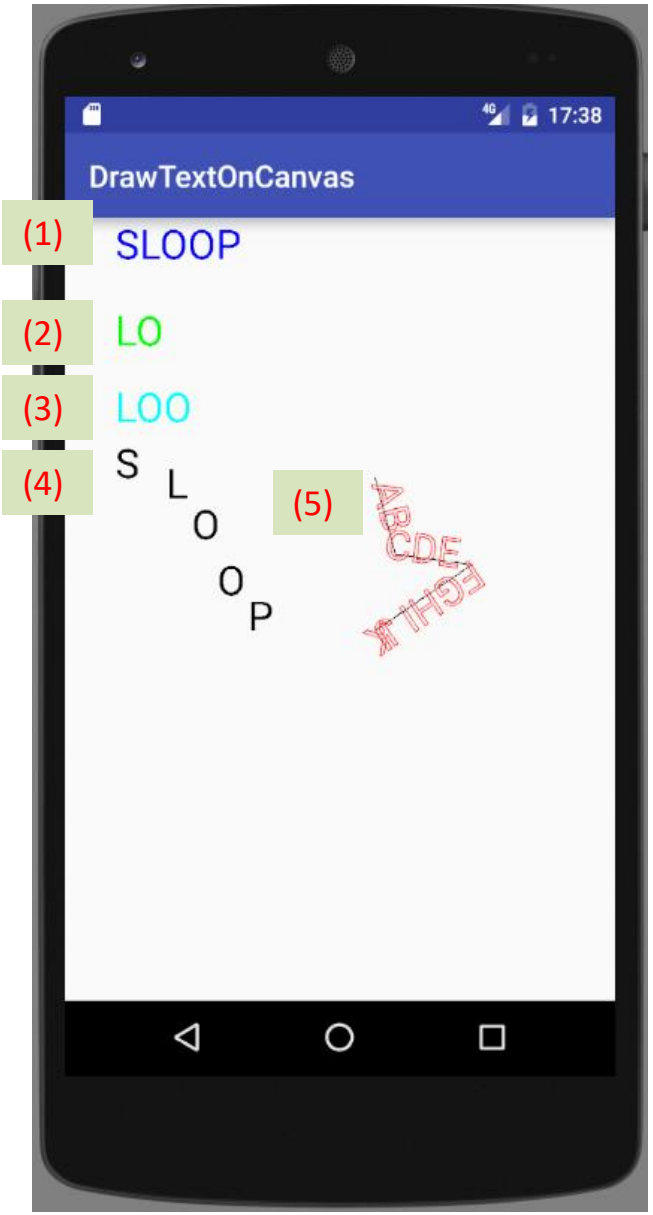
（3）指定一个路径，根据路径绘制文字

```
public void drawTextOnPath (String text, Path path, float hOffset, float vOffset,  Paint paint)
public void drawTextOnPath (char[] text, int index, int count, Path path,
                            float hOffset, float vOffset, Paint paint)
```

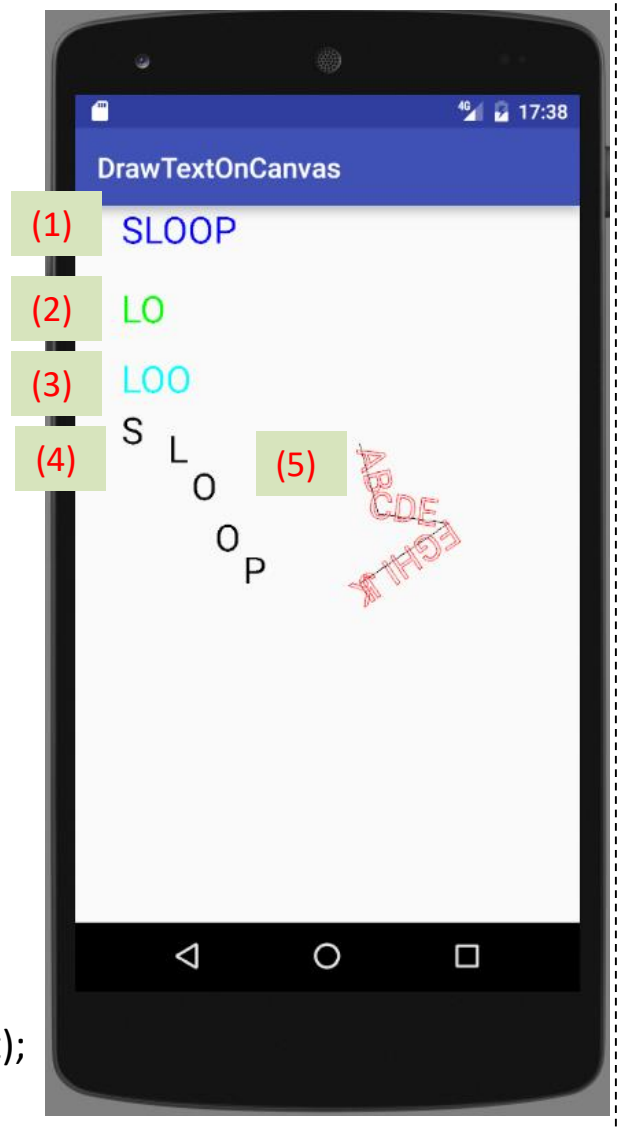
* paint.setTextSize(n)，用于drawText()时n的单位为px,而不是sp

项目DrawTextOnCanvas

```
public class MyView extends View {
    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        // 文本(要绘制的内容)
        String str = "SLOOP";
        paint.setColor(Color.BLUE); // 设置画笔颜色
        paint.setStrokeWidth(2);    // 设置画笔粗细
        paint.setTextSize(80);      // px
        // (1)参数:字符串 起始位置(x,y) 画笔
        canvas.drawText(str, 100, 80, paint);

        paint.setColor(Color.GREEN);
        // (2）参数: 字符串 开始截取位置
        //           结束截取位置 x轴坐标 y轴坐标 画笔
        canvas.drawText(str, 1, 3, 100, 250, paint);
```



```
        paint.setColor(Color.CYAN);
        char[] chars = str.toCharArray();
        // （3）参数：字符数组 起始下标 截取长度 x轴坐标 y轴坐标 画笔
        canvas.drawText(chars, 1, 3, 100, 400, paint);

        paint.setColor(Color.BLACK);
        // （4）每个字符定位显示
        canvas.drawPosText(str, new float[]{
                100, 510, // 第一个字符位置
                200, 550, // 第二个字符位置
                250, 630, // ...
                300, 740,
                360, 810}, paint);
        Path path = new Path(); //定义一条路径
        path.moveTo(610, 510); //移动到 坐标10,10
        path.lineTo(650, 660);
        path.lineTo(800, 680);
        path.lineTo(610, 810);

        paint.setStyle(Paint.Style.STROKE);
        canvas.drawPath(path, paint);
        paint.setColor(Color.RED);
        // （5）按路径显示字符串
        canvas.drawTextOnPath("AB CDE FGHI JK",   path, 10, 10, paint);
    }
}
```



- 设置画笔文本属性

setTextSize(float textSize) {}	*设置绘制文字的字号大小(sp)*
setTextAlign(Paint.Align align){}	*设置绘制文字的对齐方向*
setTextScaleX(float scaleX){}	*设置绘制文字x轴的缩放比例*
setTextSkewX(float skewX){}	*设置斜体文字，skewX为倾斜弧度*
setTypeface(Typeface typeface){}	*设置Typeface对象，即字体风格*
setUnderlineText(boolean underlineText){}	*设置带有下划线的文字效果*
setStrikeThruText(boolean strikeThruText){}	*设置带有删除线的效果*
setFakeBoldText(boolean fakeBoldText){}	*模拟实现粗体文字*
setSubpixelText(boolean subpixelText) {}	*设置是否改进文字显示(仅用于LCD)*

* setTextAlign的参数*取值：Paint.Align.CENTER, Paint.Align.LEFT, Paint.Align.RIGHT*
* setTypeface的参数*取值：Typeface.SANS_SERIF*

绘制位图

runoob

- 获得Bitmap对象

工厂模式 BitmapFactory

（1）从资源目录**raw**下获得**logo.png**的**Bitmap**对象：

```
Bitmap bitmap =
        BitmapFactory.decodeResource(mContext.getResources(),R.raw.logo);
```

或

```
InputStream is =getResources().openRawResource(R.raw.logo); //读取文件（加try语句）
bitmap = BitmapFactory.decodeStream(is);
```

（2）从资源目录**drawable**下获得**home.jpg**的**Bitmap**对象：

```
Bitmap bitmap =
        BitmapFactory.decodeResource(mContext.getResources(),R.drawable.home);
```

（3）从**assets**目录下获得**desk.png** 的**Bitmap**对象：

```
Bitmap bitmap=null;
try {
    InputStream is = mContext.getAssets().open("desk.png");
    bitmap = BitmapFactory.decodeStream(is);
    is.close();
}
catch (IOException e) {e.printStackTrace(); }
```

（4）从内存卡获得：

```
Bitmap bitmap = BitmapFactory.decodeFile("/sdcard/bitmap.png");
```

（5）从网络获得：

```
// 此处省略了获取网络输入流的代码
Bitmap bitmap = BitmapFactory.decodeStream(is); is.close();
```

（6）直接创建位图

```
Bitmap backgroundBmp
    = Bitmap.createBitmap(width,height,Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(backgroundBmp); //在backgroundBmp上画图
```

（7）用其他位图创建

```
Bitmap dstBitmap = Bitmap.createBitmap(srcBitmap, 0, 0,
                        srcBitmap.getWidth(),srcBitmap.getHeight(), matrix, true);
```

* 可以从srcBitmap中截取一块，并进行matrix变换。

- res/drawable中的文件会被压缩在apk文件中，而res/raw和res/assets中的不会。
- res/raw中的文件可以用R.id.filename访问，而res/assets中的不可以。
- res/assets则可以有子目录，res/raw不可以有。
- res/assets中的单个文件不能超过1MB，res/raw中的文件没有这个限制。

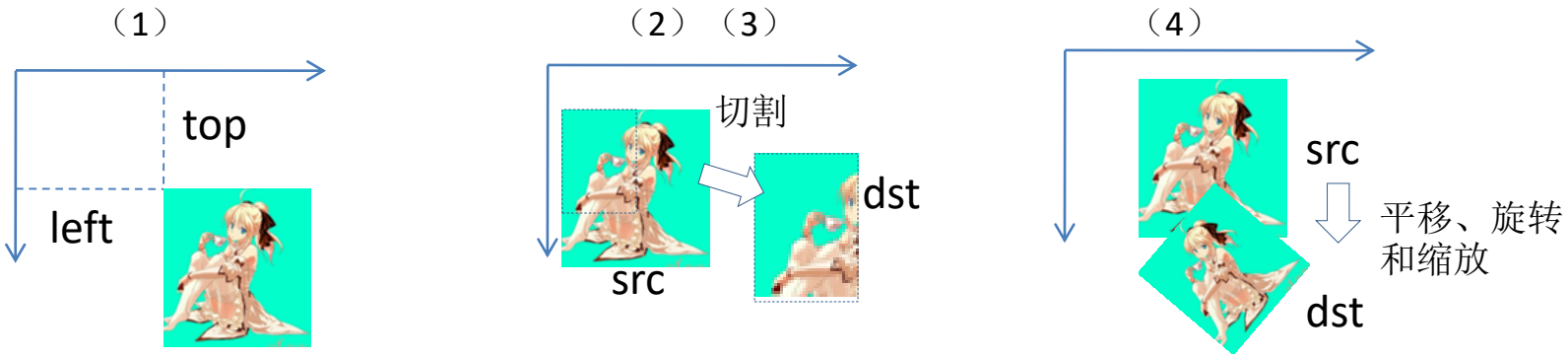
- 用drawBitmap贴图到画布上

（1）**public void drawBitmap (Bitmap bitmap, float left, float top, Paint paint)**
把bitmap贴到Canvas的坐标(left,top)处，左上角对齐该处。

（2） public void drawBitmap(Bitmap bitmap, Rect src, **Rect** dst, Paint paint)
把bitmap的一个区域src粘贴到canvas的一个区域dst，如果两个区域大小不一致，就会产生伸缩效果。当src=null时，把整个原图作为范围。

（3）public void drawBitmap (Bitmap bitmap, Rect src, **RectF** dst, Paint paint)
把bitmap的一个区域src粘贴到canvas的一个区域dst，如果两个区域大小不一致，就会产生伸缩效果。

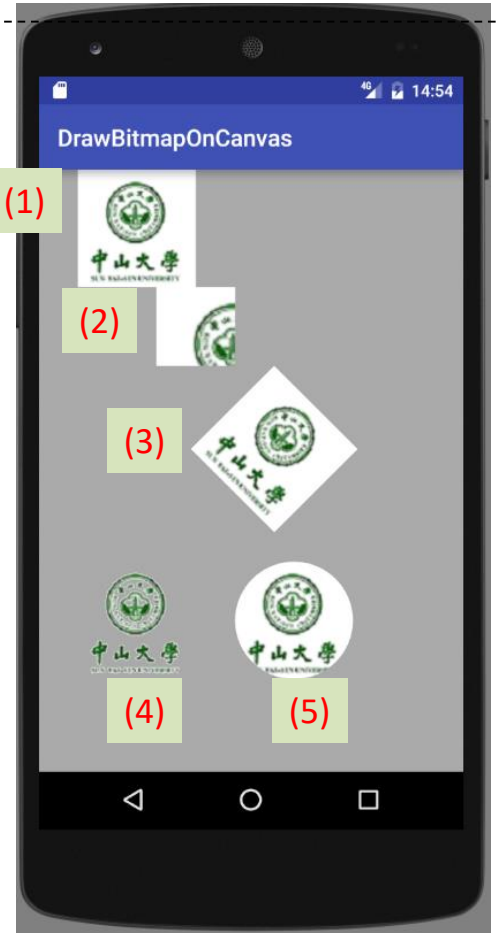
（4）public void drawBitmap (Bitmap bitmap, Matrix matrix, Paint paint)
把bitmap用matrix变换（平移、缩放、旋转）后贴到canvas中。



* public *Rect* (int left, int top, int right, int bottom) -Rect的构造器
* public *RectF* (float left, float top, float right, float bottom) -RectF的构造器

项目DrawBitmapOnCanvas

```
public class MyView extends View {
    public MyView(Context context,AttributeSet attrs){
            super(context,attrs);}
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas); Paint paint = new Paint();
        Bitmap bitmap = BitmapFactory.decodeResource(
                    this.getResources(),R.raw.sysu);
        int w = bitmap.getWidth();
        int h = bitmap.getHeight();
        //(1) 拷贝位图到画布上，开始位置为top,left
        canvas.drawBitmap(bitmap, 100, 0, paint);
        //（2）把位图的一部分拷贝到目标区域（可能会伸缩）
        canvas.drawBitmap(bitmap, new Rect(0,0,w/2,h/2),
                          new Rect(300,300,500,500), paint);
        //（3）通过Matrix进行平移和旋转后把位图复制到画布
        Matrix matrix = new Matrix();
        matrix.setTranslate(600, 500);
        matrix.preRotate(45);
        canvas.drawBitmap(bitmap, matrix, paint);
        //（4）制作透明位图
        Bitmap bitmap1=toTransparentBitmap(bitmap,0xffffffff);
        paint.setAlpha(254);   //设置paint的alpha值，不要到255
        canvas.drawBitmap(bitmap1,100,1000,paint);//有背景图片才可看到透明效果
        // (5)制作圆形位图
        canvas.drawBitmap(toRoundBitmap(bitmap),500,1000,paint);
    }
```



```java
// 把bitmap中的颜色transColor变为透明色
public Bitmap toTransparentBitmap(Bitmap bitmap,int transColor) {
    int width=bitmap.getWidth();
    int height=bitmap.getHeight();
    // 最后一个参数为isMutable
    Bitmap bitmappic = bitmap.copy(Bitmap.Config.ARGB_8888, true);
    int color;
    for (int px = 0; px < width; px++) {
        for (int py = 0; py < height; py++) {
            color = bitmappic.getPixel(px, py);
            int r=(color & 0x00FF0000)>>16;
            int g=(color & 0x0000FF00)>>8;
            int b=(color & 0x000000FF);
            int r1=(transColor & 0x00FF0000)>>16;
            int g1=(transColor & 0x0000FF00)>>8;
            int b1=(transColor & 0x000000FF);
            // 采用近似方法而不是color == transColor
            if(Math.abs(r1-r)+Math.abs(g1-g)+Math.abs(b1-b)<80)
                bitmappic.setPixel(px, py, 0x00000000);
        }
    }
    return bitmappic;
}
```

```
    // 把一个位图变成圆形位图
    public Bitmap toRoundBitmap(Bitmap bitmap) {
        //圆形图片宽高
        int width = bitmap.getWidth();
        int height = bitmap.getHeight();
        //正方形的边长
        int r = (width>height)?height:width; //取最短边做边长
        //构建一个bitmap作为背景图
        Bitmap backgroundBmp = Bitmap.createBitmap(width,
                height, Bitmap.Config.ARGB_8888);//确定canvas的长宽
        Canvas canvas = new Canvas(backgroundBmp);//背景色默认为白色
        Paint paint = new Paint();
        paint.setAntiAlias(true);
        canvas.drawCircle(r/2, r/2, r/2, paint);  //画了一个圆
        //设置当两个图形相交时的模式，SRC_IN为取SRC图形相交的部分，多余的将被去掉
        RectF rect = new RectF(0, 0, r, r);
        paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
        canvas.drawBitmap(bitmap, null, rect, paint);//与背景叠加
        return backgroundBmp;
    }
}
```

Matrix变换

参考

- 概述

Matrix可以用于图像的变换：平移（translate）、旋转（rotate）、缩放（scale）、错切或倾斜（skew），也可以进行任意的3×3矩阵变换。

```
public void setTranslate(float dx, float dy);
public void setRotate(float degrees);
public void setScale(float sx, float sy);
public void setSkew(float kx, float ky);
public void setScale(float sx, float sy, float px, float py);
public void setRotate(float degrees, float px, float py);
public void setSkew(float kx, float ky, float px, float py);
```

px=pivotX,
py=pivotY

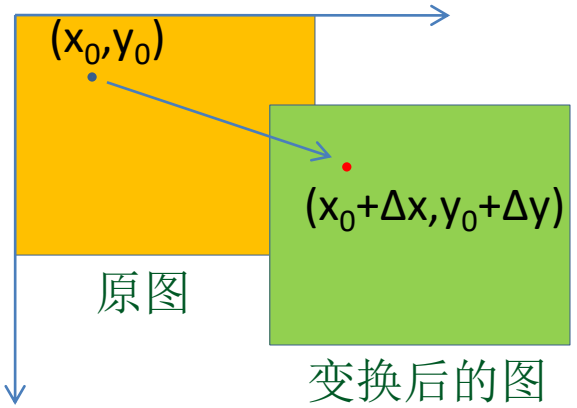
变换(旋转、缩放和倾斜) 默认情况是围绕原点(0，0)点进行，还可以围绕一个中心点(px，py)来进行。这些变换都是针对要绘制的图像进行的，其实它们也可以从Canvas变换后得到。

```
Matrix  matrix = new Matrix();
canvas.drawBitmap(bitmap, matrix, new Paint());  //左上角在原点(0,0)
matrix.setTranslate(100, 1000);
canvas.drawBitmap(bitmap, matrix, new Paint());
```

• 变换的原理

变换

（1）平移： 点(x_0,y_0)平移到(x,y)



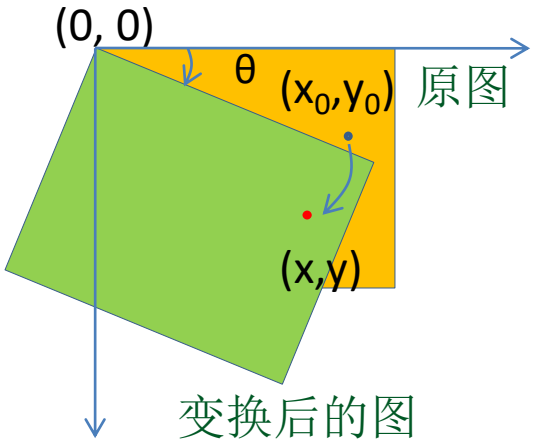
$$\Delta x = x - x_0$$

$$\Delta y = y - y_0$$

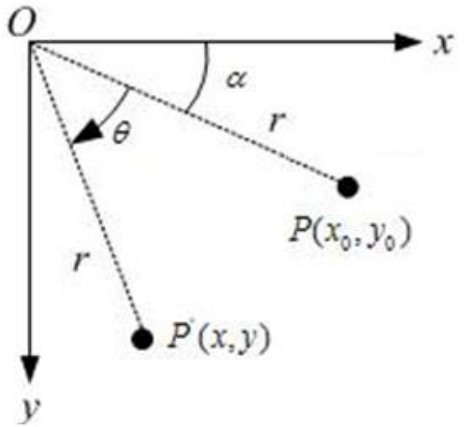
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

public void setTranslate(float dx, float dy)

（2）旋转： 点(x_0,y_0)绕原点顺时针方向旋转至(x,y)



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$



$$x_0 = r\cos\alpha$$

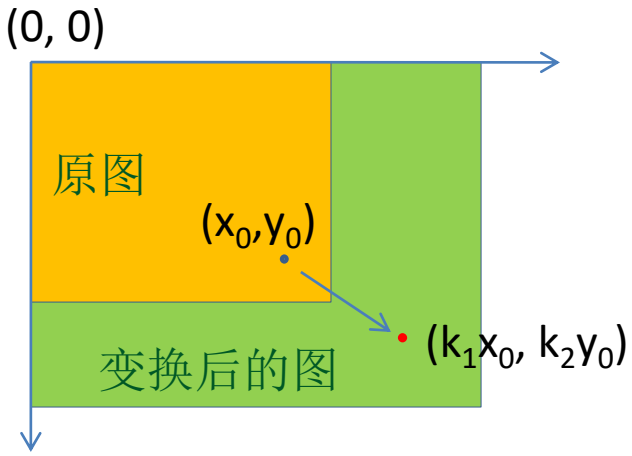
$$y_0 = r\sin\alpha$$

$$x = r\cos(\alpha+\theta) = r\cos\alpha\cos\theta - r\sin\alpha\sin\theta = x_0\cos\theta - y_0\sin\theta$$

$$y = r\sin(\alpha+\theta) = r\sin\alpha\cos\theta + r\cos\alpha\sin\theta = y_0\cos\theta + x_0\sin\theta$$

public void setRotate(float degrees)

（3）缩放变换

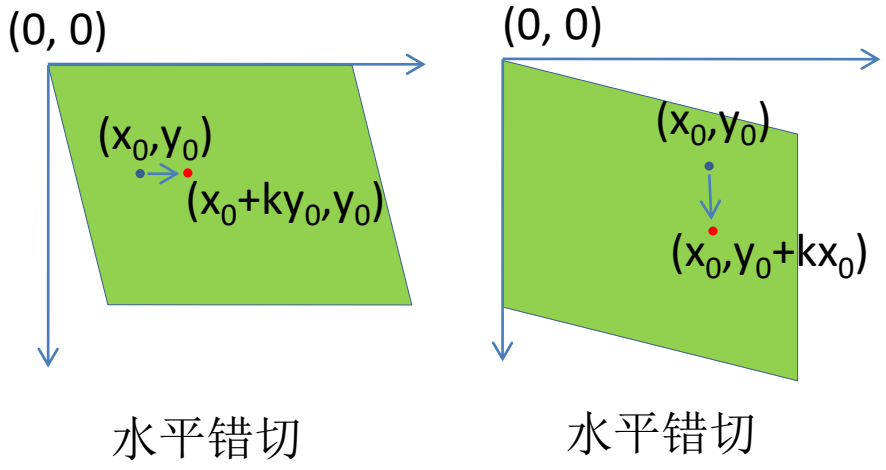


$$x = k_1 x_0$$
$$y = k_2 y_0$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

public void setScale(float sx, float sy)

（4）错切变换（倾斜变换）



$$x = x_0 + ky_0$$
$$y = y_0$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & k & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

$$x = x_0$$
$$y = k x_0 + y_0$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ k & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

同时变换

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & k_1 & 0 \\ k_2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

public void setSkew(float kx, float ky)

（5）另类旋转变换

```
public void setSinCos(float sinValue, float cosValue);
```

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

sinValue：对应图中的sin值，cosValue：对应cos值

```
public void setSinCos(float sinValue, float cosValue,
                                    float px, float py) ;
```

px和py见后面的轴心变换

（6）并置变换

```
public boolean setConcat(Matrix a,Matrix b);
```

将当前matrix的值变为a和b的乘积。

Matrix的其他方法见附录。

- 组合matrix矩阵

参考　　参考

$$\begin{pmatrix} MSCALE\_X & MSKEW\_X & MTRANS\_X \\ MSKEW\_Y & MSCALE\_Y & MTRANS\_Y \\ MPERSP\_0 & MPERSP\_1 & MPERSP\_2 \end{pmatrix}$$

scale是缩放，skew是错切，trans是平移，persp代表透视

当最后一行为0、0、1时，其矩阵变换进行仿射变换，即是二维坐标到二维坐标的线性变换，并保持二维图形的“平直性”和“平行性”。**保持平直性**是指图像变换后直线还是直线不会打弯，圆弧还是圆弧，**保持平行性**指保持二维图形间的相对位置关系不变，平行线还是平行线，而直线上点的位置顺序不变。

每个set方法都会把上个set方法的效果清除掉，例如，依次调用了setSkew和setTranslate，最终只有setTranslate会起作用，那么如何才能将两种效果复合在一起呢？主要有两种方法：

preXXXX:　　以pre开头，例如preTranslate
postXXXX:　　以post开头，例如postScale

pre和pro表示乘号在之前还是之后，也就是原变换是放在本变换之前还是之后。

以下语句实现先缩放再移动（按缩放比）

```
Matrix matrix = new Matrix();
matrix.setTranslate(100,1000);
matrix.postScale(0.5f,0.5f);   // 乘号放在Scale之后，即先缩放再平移
canvas.drawBitmap(bitmap, matrix, paint);
```

这里matrix乘了一个scale矩阵，换算成数学式如下：

$$\begin{bmatrix}0.5 & 0 & 0\\ 0 & 0.5 & 0\\ 0 & 0 & 1\end{bmatrix} * \begin{bmatrix}1 & 0 & 100\\ 0 & 1 & 1000\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0.5 & 0 & 50\\ 0 & 0.5 & 500\\ 0 & 0 & 1\end{bmatrix}$$

$X=X_0*0.5+50$

$Y=Y_0*0.5+500$

上面画红线的两条语句用以下语句代替：

```
matrix.setScale(0.5f,0.5f);
matrix.preTranslate(100,1000); // 乘号放在Translate之前，即后Translate
```

下面的矩阵变换可以用什么语句得到？

$$\begin{bmatrix}1 & 0 & 100\\ 0 & 1 & 1000\\ 0 & 0 & 1\end{bmatrix} * \begin{bmatrix}0.5 & 0 & 0\\ 0 & 0.5 & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0.5 & 0 & 100\\ 0 & 0.5 & 1000\\ 0 & 0 & 1\end{bmatrix}$$

```
matrix.setTranslate(100,1000);
matrix.   ?   (0.5f,0.5f);
```

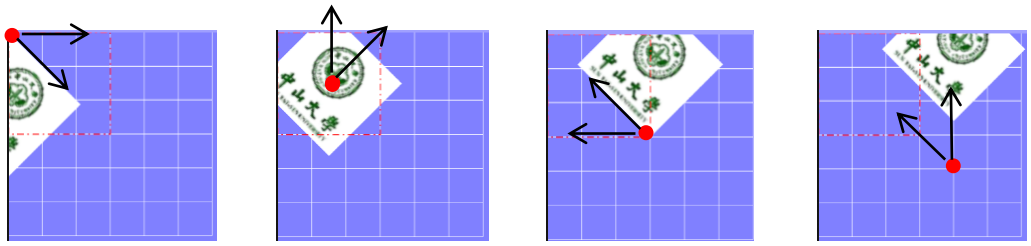
- 轴心变换

```
matrix.setRotate(degrees,px,py);
```

等同

```
matrix.setTranslate(px,py);
matrix.preRotate(deg);
matrix.preTranslate(-px,-py);
```

```
matrix.setRotate(45f,0,0);
matrix.setRotate(45f,150,150);
matrix.setRotate(45f,300,300);
matrix.setRotate(45f,400,400);
```



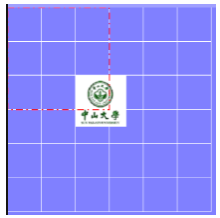
(0, 0) (150, 150) (300, 300) (400, 400)

```
matrix.setScale(sx,sy,px,py);
matrix.setSkew(kx,ky,px,py);
```

与setRotate类似

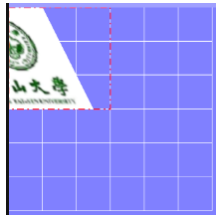
```
matrix.setScale(0.5f,0.5f,400,400);
```

```
matrix.setTranslate(400, 400);
matrix.preScale(0.5f,0.5f);
matrix.preTranslate(-400, -400);
```

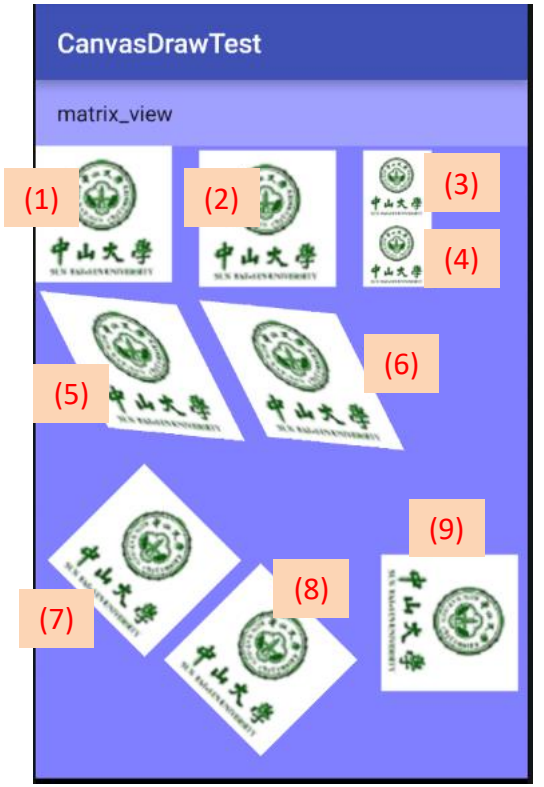


```
matrix.setSkew(0.5f, 0f,400, 400);
```

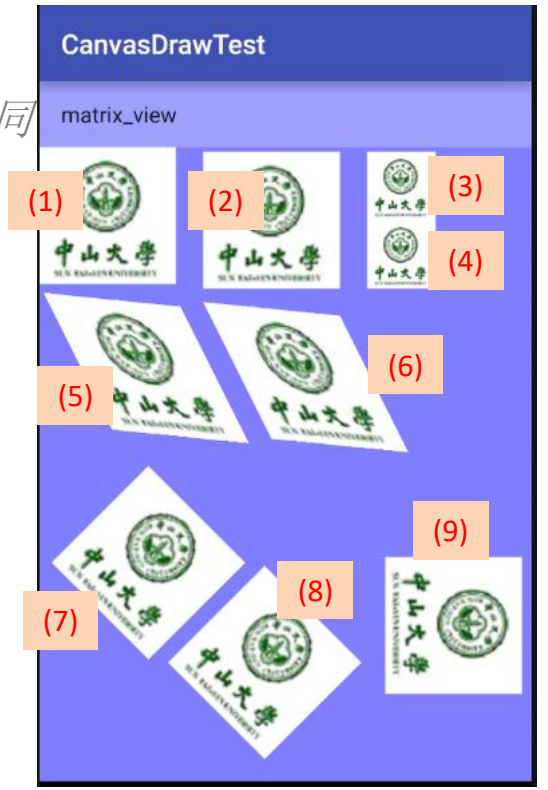
```
matrix.setTranslate(400, 400);
matrix.preSkew(0.5f,0f);
matrix.preTranslate(-400, -400);
```



```
public class MatrixView extends View {
    Context context;
    public MatrixView(Context context, AttributeSet attrs) {
        super(context, attrs); this.context = context;
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(), R.raw.sysu);
        int w = bitmap.getWidth();  //300px
        int h = bitmap.getHeight(); //300px
        Paint paint = new Paint();
        canvas.drawRGB(128, 128, 255);
        //(1)原图
        canvas.drawBitmap(bitmap, 0,0, paint);
        //(2)平移
        Matrix matrix = new Matrix();
        matrix.setTranslate(360, 10);//dx,xy
        canvas.drawBitmap(bitmap, matrix, paint);
        //(3)先位移，再缩放
        matrix.setTranslate(720, 10);//会自动复位
        matrix.preScale(0.5f,0.5f);
        canvas.drawBitmap(bitmap, matrix, paint);
        //(4)先缩放，再位移(按缩放比例位移)
        matrix.setTranslate(1440, 320);
        matrix.postScale(0.5f,0.5f);
        canvas.drawBitmap(bitmap, matrix, paint);
```

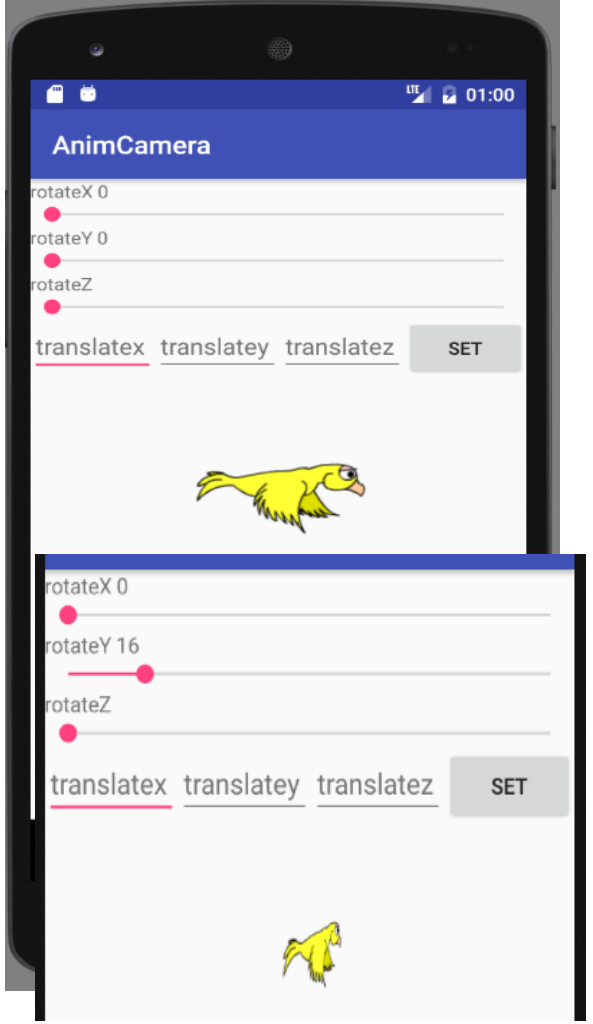


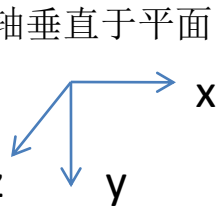
```
        //(5)先平移再错切
        //matrix.setSkew(0.5f,0.1f);    // 这两句
        //matrix.postTranslate(10,320);// 与下面两句功能相同
        matrix.setTranslate(10,320);
        matrix.preSkew(0.5f,0.1f);
        canvas.drawBitmap(bitmap,matrix,paint);
         //(6)先错切再平移
        //matrix.setSkew(0.5f,0.1f);    // 这两句
        //matrix.preTranslate(200,320);// 与下面两句功能相同
        matrix.setTranslate(200,320);
        matrix.postSkew(0.5f,0.1f);
        canvas.drawBitmap(bitmap,matrix,paint);

        //(7)先平移再旋转
        matrix.setRotate(45);
        matrix.postTranslate(240,700);
        canvas.drawBitmap(bitmap, matrix, paint);
        //(8)先旋转再平移(按旋转后的坐标平移)
        matrix.setRotate(45);
        matrix.preTranslate(1000,300);
        canvas.drawBitmap(bitmap, matrix, paint);
        //（9）先移位顺时针再旋转θ角度
        matrix.setSinCos(1, 0, w/2, h/2);//sin θ=1,cos θ=0,px,py
        matrix.postTranslate(760,900);
        canvas.drawBitmap(bitmap, matrix, paint);
    }
}
```



Camera变换

通过Camera类可以产生用于3D变换的Matrix。






```
Camera mCamera= new Camera();

mCamera.save();
mCamera.rotateX(degreeX);//度数(浮点数)
mCamera.rotateY(degreeY);
mCamera.rotateZ(degreeZ);
mCamera.translate(translateX, translateY, translateZ);
Matrix matrix = new Matrix();
mCamera.getMatrix(matrix);
mCamera.restore();

BitmapDrawable bitmapDrawable = (BitmapDrawable)
            getResources().getDrawable(R.drawable.sprite);
Bitmap tempBitmap = bitmapDrawable.getBitmap();
Bitmap newBitmap = null;
newBitmap = Bitmap.createBitmap(tempBitmap, 0, 0, tempBitmap.getWidth(),
                                tempBitmap.getHeight(), matrix, true);
imgView.setImageBitmap(newBitmap);
```

```
public class MainActivity extends AppCompatActivity {
    Camera mCamera;
    private ImageView iv_content;
    private SeekBar sb_x, sb_y, sb_z;
    private EditText et_x, et_y, et_z;
    private TextView tv_rotatex, tv_rotatey, tv_rotatez;
    private float rotateX, rotateY, rotateZ;
    private float translateX, translateY, translateZ;
    private Button btn_set;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mCamera = new Camera();
        initViews();      registerListeners();
    }
    private void initViews() {
        sb_x = (SeekBar) findViewById(R.id.sb_rotatex);       sb_x.setMax(100);
        sb_y = (SeekBar) findViewById(R.id.sb_rotatey);       sb_y.setMax(100);
        sb_z = (SeekBar) findViewById(R.id.sb_rotatez);       sb_z.setMax(100);
        et_x = (EditText) findViewById(R.id.et_x);
        et_y = (EditText) findViewById(R.id.et_y);
        et_z = (EditText) findViewById(R.id.et_z);
        tv_rotatex = (TextView) findViewById(R.id.tv_rotatex);
        tv_rotatey = (TextView) findViewById(R.id.tv_rotatey);
        tv_rotatez = (TextView) findViewById(R.id.tv_rotatez);
        btn_set = (Button) findViewById(R.id.btn_set);
        iv_content = (ImageView) findViewById(R.id.iv_content);
    }
```

```
private void registerListeners() {
        sb_x.setOnSeekBarChangeListener(onSeekBarChangeListener);
        sb_y.setOnSeekBarChangeListener(onSeekBarChangeListener);
        sb_z.setOnSeekBarChangeListener(onSeekBarChangeListener);

        btn_set.setOnClickListener(onClicker);
    }

    View.OnClickListener onClicker = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            translateX = getFloat(et_x.getText().toString());
            translateY = getFloat(et_y.getText().toString());
            translateZ = getFloat(et_z.getText().toString());

            refreshImage();
        }
    };

    private float getFloat(String txt) {
        try {
            if ("".equals(txt) || txt == null) {
                return 0;
            } else {
                return Float.parseFloat(txt);
            }
        } catch (Exception e) {
            return 0;
        }
    }
```

```
SeekBar.OnSeekBarChangeListener onSeekBarChangeListener =
    new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser){
            if (seekBar == sb_x) {
                tv_rotatex.setText("rotateX " + progress + "");
                rotateX = progress * 360f / 100f;
            } else if (seekBar == sb_y) {
                tv_rotatey.setText("rotateY " + progress + "");
                rotateY = progress * 360f / 100f;
            } else if (seekBar == sb_z) {
                tv_rotatez.setText("rotateZ " + progress + "");
                rotateZ = progress * 360f / 100f;
            }
            refreshImage();
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {   }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {  }
    };
```

```
private void refreshImage() {
        Matrix matrix = new Matrix();

        mCamera.save();
        mCamera.rotateX(rotateX);
        mCamera.rotateY(rotateY);
        mCamera.rotateZ(rotateZ);
        mCamera.translate(translateX, translateY, translateZ);
        mCamera.getMatrix(matrix);
        mCamera.restore();

        BitmapDrawable bitmapDrawable =
                        (BitmapDrawable) getResources().getDrawable(R.drawable.sprite1);
        Bitmap temBitmap = bitmapDrawable.getBitmap();
        Bitmap bitmap = null;
        try {
            // 经过矩阵转换后的图像宽高有可能不大于0，此时会抛出IllegalArgumentException
            bitmap = Bitmap.createBitmap(temBitmap, 0, 0, temBitmap.getWidth(),
                                                    temBitmap.getHeight(), matrix, true);
        } catch (IllegalArgumentException iae) {
            iae.printStackTrace();
        }
        if (bitmap != null) {
            iv_content.setImageBitmap(bitmap);
        }
    }

}
```

main_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/ll_seek"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tv_rotatex"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="rotateX"/>

        <SeekBar
            android:id="@+id/sb_rotatex"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <TextView
            android:id="@+id/tv_rotatey"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="rotateY"/>
```

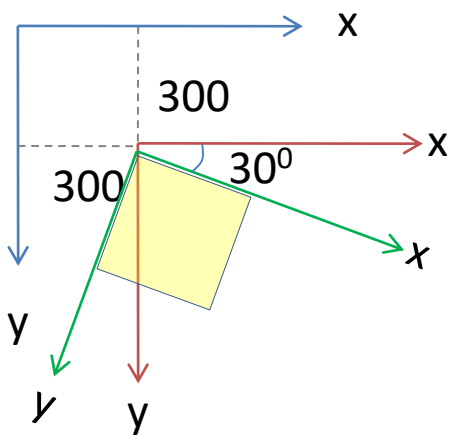
```
<SeekBar
    android:id="@+id/sb_rotatey"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/tv_rotatez"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="rotateZ"/>

<SeekBar
    android:id="@+id/sb_rotatez"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:id="@+id/et_x"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="translatex"/>
```

```
        <EditText
            android:id="@+id/et_y"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="translatey"/>

        <EditText
            android:id="@+id/et_z"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="translatez"/>

        <Button
            android:id="@+id/btn_set"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="set"/>
    </LinearLayout>
</LinearLayout>
<ImageView
    android:id="@+id/imgView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:src="@drawable/sprite"/>
</RelativeLayout>
```

Canvas变换

- Canvas可以进行平移(translate)、旋转(rotate)、缩放(scale)和错切(skew)变换。这些变换会引起整个坐标系的改变，使得之后对Canvas的绘图操作都是在变换后的坐标系进行。
- 多次Canvas变换会叠加在一起，例如，Canvas平移后旋转会使坐标系的原点发生平移并旋转。

```
paint.setColor(Color.argb(181, 255,255, 0));
canvas.translate(300, 300);                    // 原点移至(200,200)
canvas.rotate(30);                             // 坐标系绕原点旋转30度
canvas.drawRect(new Rect(0, 0, 300, 300),paint);
```



参考1 参考2 参考3 参考4

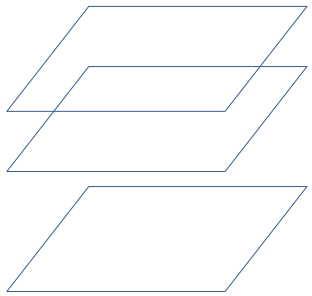
- 如果希望保持Canvas上渲染的独立性，有哪些方法？

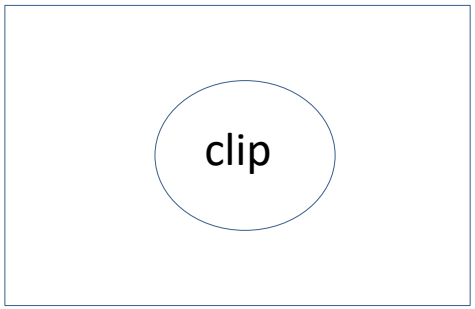
  方法1、采用Canvas的方法save和restore：通过Canvas的save方法保存当前Canvas的设置，然后用新的设置在Canvas上进行绘图，再用restore恢复Canvas原来的设置。

  方法2、采用Canvas的savetLayer可以进行多层绘图，每层绘图都可以独立进行。见saveLayer一节。

  方法3、通过Canvas的多种Clip方法还可以从Canvas上剪切出一块区域，针对这块区域进行绘图时会与Canvas的其他部分隔离，渲染时超出该区域的内容不会显示出来。

```
…
canvas.save();
…
canvas.restore();
…
```

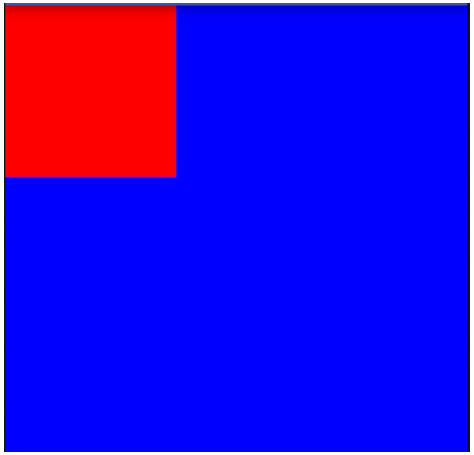




方法1. save和restore　　方法2. setLayer　　方法3. clip

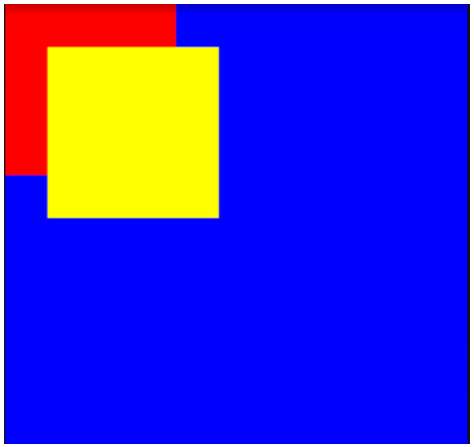
- **Canvas变换举例：**

（1）
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint()
    paint.setColor(Color.RED);
    canvas.drawColor(Color.BLUE);
    canvas.drawRect(new Rect(0, 0, 400, 400), paint);
    ...
}
```

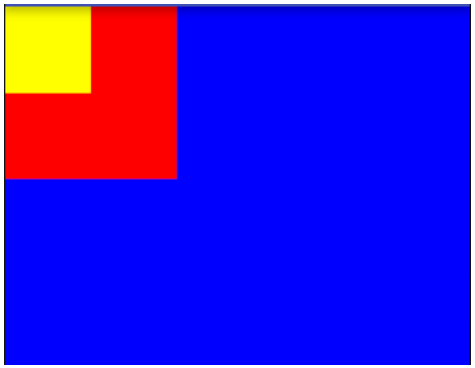


* 后面均在最后一条语句后加入一组语句

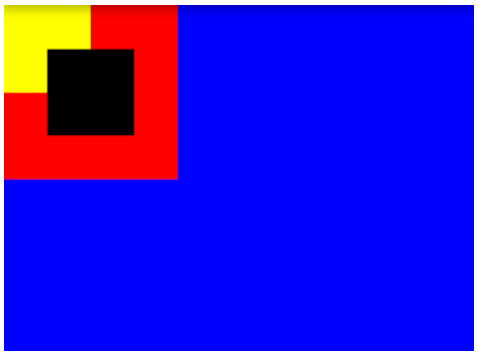
（2）
```
paint.setColor(Color.YELLOW);
canvas.translate(100, 100);
canvas.drawRect(new Rect(0, 0, 400, 400), paint);
```



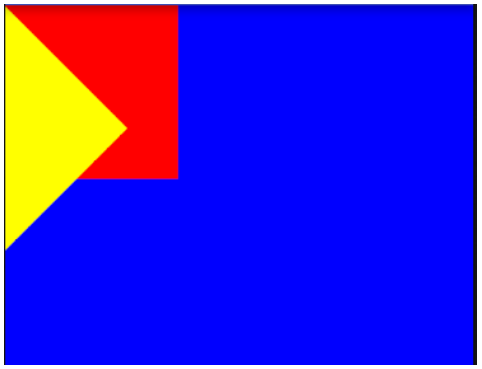
（3）
```
canvas.scale(0.5f, 0.5f);
paint.setColor(Color.YELLOW);
canvas.drawRect(new Rect(0, 0, 400, 400), paint);
```



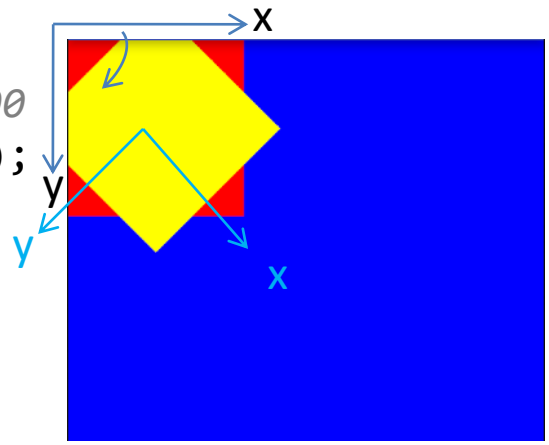
（4）
```
// 保存画布状态
canvas.save();
canvas.scale(0.5f, 0.5f);
paint.setColor(Color.YELLOW);
canvas.drawRect(new Rect(0, 0, 400, 400),paint);
// 画布状态回滚
canvas.restore();
canvas.scale(0.5f, 0.5f, 200, 200); //原点移至(100,100)
paint.setColor(Color.BLACK);
canvas.drawRect(new Rect(0, 0, 400, 400), paint);
```



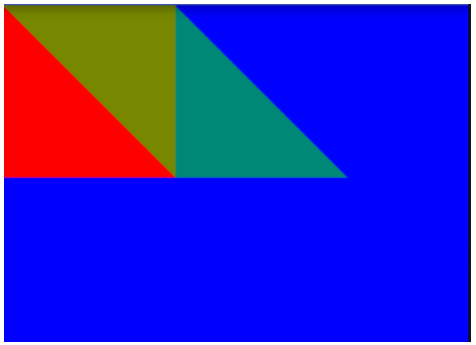
（5）
```
paint.setColor(Color.YELLOW);
canvas.rotate(45); //顺时钟绕原点旋转45度
canvas.drawRect(new Rect(0, 0, 400, 400), paint);
```



（6）
```
paint.setColor(Color.YELLOW);
canvas.rotate(45,200,200); //旋转原点移动到200,200
canvas.drawRect(new Rect(0, 0, 400, 400), paint);
```

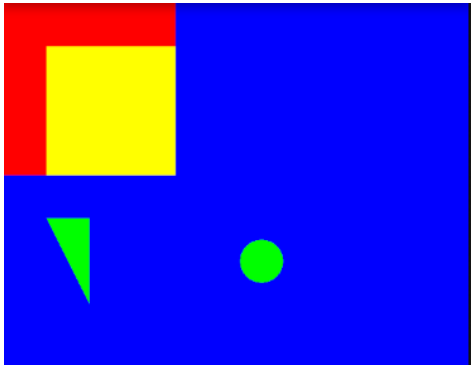


（7）
```
canvas.skew(1, 0);  //x方向错切，y方向错切
paint.setColor(0x8800ff00);
canvas.drawRect(new Rect(0, 0, 400, 400), paint);
```



（8）
```
canvas.skew(0, 1); //x方向错切，y方向错切
paint.setColor(0x8800ff00);
canvas.drawRect(new Rect(0, 0, 400, 400), paint);
```

（9）
```
Rect rect = new Rect(100, 100, 400, 400);
canvas.save();
canvas.clipRect(rect);
canvas.drawColor(Color.YELLOW);
canvas.restore();
Path path=new Path();
path.moveTo(100,500);
path.lineTo(200,500);
path.lineTo(200,700);
path.lineTo(100,500);
path.addCircle(600,600,50, Path.Direction.CCW);
canvas.clipPath(path);
canvas.drawColor(Color.GREEN);
```



* 关于clip，见下一节

Canvas的clip方法

- Canvas的clip方法用于从Canvas上剪裁出一块区域，绘制的内容只有在裁剪区域才能显示出来。
- canvas.save()和canvas.restore()不仅对matrix有效，同样对clip有类似的效果。
- Canvas提供了三种裁剪区域的方式：裁剪出一个矩形（ clipRect ），裁剪出一个路径（ clipPath ），剪裁出一个Region（clipRegion）。
- Region用于组合若干区域。与clipRect和clipPath不同，canvas的matrix对clipRegion没有影响。由于clipPath可以用于类似clipRegion的功能，clipRegion已被弃用。
- 剪裁的源图(Src)可以叠加到上次剪裁的目标图(Dst)上，有六种叠加方法：

Region.Op.DIFFERENCE	显示Src中不同于Dst的部分
Region.Op.REPLACE	只显示Src
Region.Op.UNION	Src和Dst的并集
Region.Op.XOR	Src和Dst中的非重叠部分
Region.Op.INTERSECT	Src和Dst交集（重叠部分）
Region.Op.REVERSE_DIFFERENCE	显示Src中不同于Dst的部分(默认)

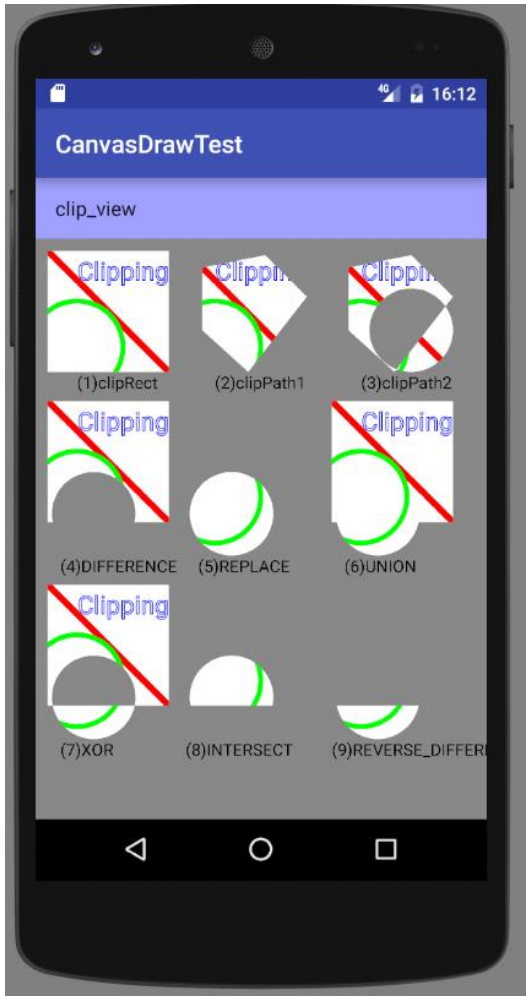
- 可用的clip方法

```
boolean clipPath(Path path)
boolean clipPath(Path path, Region.Op op)

boolean clipRect(Rect rect, Region.Op op)
boolean clipRect(RectF rect, Region.Op op)
boolean clipRect(int left, int top, int right, int bottom)
boolean clipRect(float left, float top, float right, float bottom)
boolean clipRect(RectF rect)
boolean clipRect(float left, float top, float right, float bottom, Region.Op op)
boolean clipRect(Rect rect)

boolean clipRegion(Region region)
boolean clipRegion(Region region, Region.Op op)
```

已过时

- 在clip范围内draw

```
private void drawScene(Canvas canvas) {
    canvas.drawColor(Color.WHITE);

    mPaint.setColor(Color.RED);
    mPaint.setStrokeWidth(16);
    canvas.drawLine(10, 10, 320, 320, mPaint);

    mPaint.setColor(Color.GREEN);
    mPaint.setStrokeWidth(12);
    mPaint.setStyle(Paint.Style.STROKE);
    canvas.drawCircle(80, 240,110, mPaint);

    mPaint.setColor(Color.BLUE);
    mPaint.setStrokeWidth(2);
    mPaint.setTextSize(60);
    canvas.drawText("Clipping", 300, 80, mPaint);
}
```



- clipRect

```
canvas.clipRect(10, 10, 300, 300);
drawScene(canvas);
```

window背景是灰色的

- clipPath（1）

```
Path path = new Path();
path.moveTo(50,50);
path.lineTo(200,20);
path.lineTo(300,120);
path.lineTo(160,300);
path.lineTo(50,200);
path.close();
canvas.clipPath(path);
drawScene(canvas);
```

window背景是灰色的

- clipPath（2）

```
path.addCircle(200f,200f,100f, Path.Direction.CCW);
canvas.clipPath(path);//XOR
drawScene(canvas);
```

在原来path上增加了一个圆

window背景是灰色的

CCW-counterclock wise

- 两个区域叠加

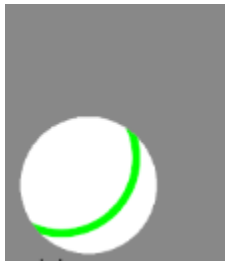
在剪裁Src(圆形)时有六种方法叠加上次剪裁Dst(矩形)的结果。

只采用Dst(矩形)剪裁

- DIFFERENCE　　*其它操作类似*

```
canvas.clipRect(10, 10, 300, 300);                    // Dst（矩形）
canvas.clipPath(path, Region.Op.DIFFERENCE);          // Src（圆形）
drawScene(canvas);
```

DIFFERENCE: 显示Dst(矩形)中不同于Src(圆形)的部分



REPLACE : 只显示Src(圆形)

UNION : Src(圆形)和Dst(矩形)的并集

XOR: 显示Src(圆形)和Dst(矩形)中的非重叠部分

INTERSECT: 显示Src(圆形)和Dst(矩形)中的重叠部分

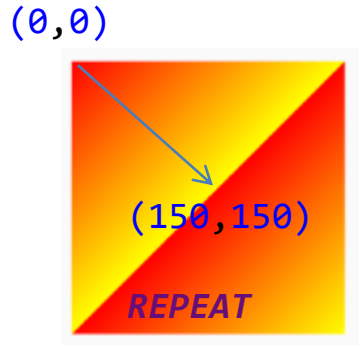
REVERSE_DIFFERENCE : 显示Src(圆形)中不同于Dst (矩形)的部分

设置着色器 (setShader)

参考1 参考2

（1）线性渐变

```
Shader shader = new LinearGradient(0, 0, 150, 150,
        Color.RED, Color.YELLOW, Shader.TileMode.REPEAT);
paint.setShader(shader);
canvas.drawRect(0, 0, 300, 300, paint);
```



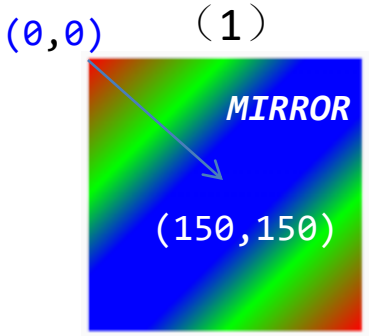
（1）

（2）多段线性渐变

```
Shader shader = new LinearGradient(0, 0, 150, 150,
        new int[]{Color.RED, Color.GREEN, Color.BLUE},
        new float[]{0, 0.5F, 0.8F},
        Shader.TileMode.MIRROR);
```

positions 取值: 0~1

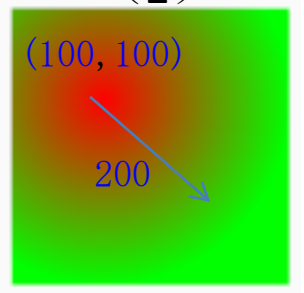
剩余部分(0.8~1)
用最后颜色填充



（2）

（3）辐射渐变（径向渐变）

```
Shader shader = new RadialGradient(100, 100, 200,
        Color.RED, Color.GREEN, Shader.TileMode.CLAMP);
```



当填充区域大于Shader时，用Shader.TileMode确定空白部分的填充方法：
CLAMP(重复边沿填充)、REPEAT(重复)、 MIRROR(镜像)

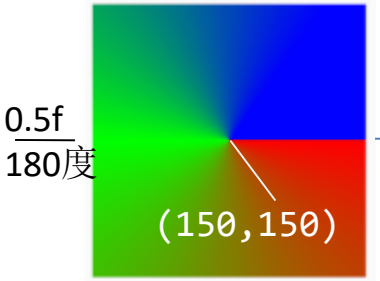
（3）

项目：CanvasDrawTest
shader_view

（4）扫描渐变

```
Shader shader = new SweepGradient(150, 150, new int[]
        {Color.RED, Color.GREEN, Color.BLUE},
         new float[]{0f, 0.5f, 0.85f});
```

positions



（4）

position取值范围为[0,1]，对应0~360度，取值null时，自动设置等间距。

（5）组合渐变

```
Bitmap bitmap = ((BitmapDrawable)
getResources().getDrawable(R.drawable.music1)).getBitmap();
Shader bitmapShader = new BitmapShader(bitmap, Shader.TileMode.MIRROR,
Shader.TileMode.MIRROR);
Shader linearGradient = new LinearGradient(0, 0, 100, 100, new int[] {
        Color.WHITE, Color.LTGRAY, Color.TRANSPARENT, Color.GREEN },
        null, Shader.TileMode.MIRROR); //平铺效果为镜像
ComposeShader shader = new ComposeShader(bitmapShader,
        linearGradient,PorterDuff.Mode.DARKEN);
```

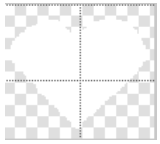
R.drawable.music1

（5）

（6）组合渐变

```
bitmap = ((BitmapDrawable)
getResources().getDrawable(R.drawable.heart)).getBitmap();
int bitmapWidth = bitmap.getWidth();
int bitmapHeight = bitmap.getHeight();
canvas.saveLayer(0, 0, bitmapWidth, bitmapHeight, null,
                                                  Canvas.ALL_SAVE_FLAG);
bitmapShader = new BitmapShader(bitmap, Shader.TileMode.CLAMP,
                                                  Shader.TileMode.CLAMP);
paint.setShader(bitmapShader);
canvas.drawRect(0, 0, bitmapWidth, bitmapHeight, paint);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.MULTIPLY));
linearGradient = new LinearGradient(0, 0, bitmapWidth, bitmapHeight,
                            Color.GREEN, Color.BLUE, Shader.TileMode.CLAMP);
paint.setShader(linearGradient);
canvas.drawRect(0, 0,  bitmapWidth,  bitmapHeight, paint);
paint.setXfermode(null);
canvas.restore();
```

* 其他见“setShadowLayer”和“Matrix”



Heart.png
中间白色
其余透明色

（6）

设置遮罩滤镜（setMaskFilter）

参考1 参考2

paint的方法setMaskFilter用于图像的滤镜变换，它传入的参数为一个MaskFilter对象。MaskFilter有两个子类BlurMaskFilter和EmbossMaskFilter，前者为模糊遮罩滤镜而后者为浮雕遮罩滤镜。

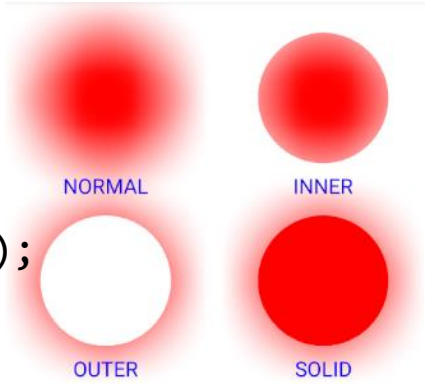
（1）BlurMaskFilter(float radius, BlurMaskFilter.Blur style)

radius *模糊半径*

style *模糊样式，取值NORMAL、INNER、OUTER、SOLID*

例：
```
BlurMaskFilter blur=new BlurMaskFilter(10,NORMAL);
    paint.setMaskFilter(blur);
```

* 可能要加入“setLayerType(*LAYER_TYPE_SOFTWARE*, null)”关闭硬件加速才能显示出效果。



模糊遮罩滤镜

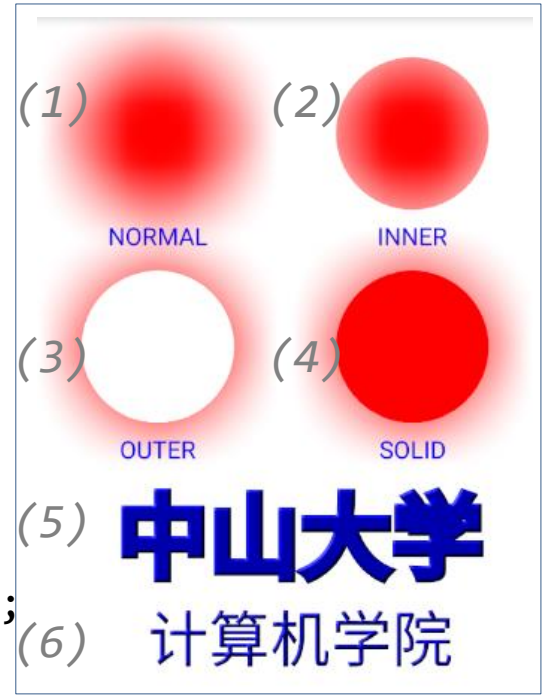
（2）EmbossMaskFilter(float[] direction, float ambient, float specular, float radius)

direction	3个元素的float数组，用来指定光源的方向
ambient	定义环境光亮度，取值0~1。
specular	定义镜面反射系数。
radius	模糊半径。

例：
```
EmbossMaskFilter emboss=new EmbossMaskFilter(
                        new float[]{10, 10, 10}, 0.1f, 5f, 5f);
   paint.setMaskFilter(emboss);
```

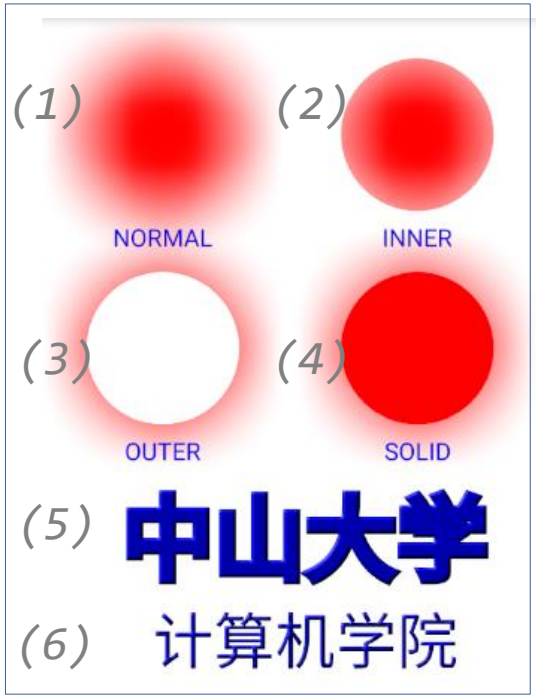
浮雕遮罩滤镜

```
public class MaskFilterView extends View {
    Context context;
    public MaskFilterView(Context context, AttributeSet attrs) {
        super(context, attrs); this.context = context;
    }
    @SuppressLint("NewApi")     //新功能不报错
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        //关闭硬件加速
        setLayerType(LAYER_TYPE_SOFTWARE, null);
        Paint paint = new Paint(ANTI_ALIAS_FLAG);
        paint.setColor(Color.RED);
        paint.setStyle(Paint.Style.FILL_AND_STROKE);

        Paint textPaint = new TextPaint(ANTI_ALIAS_FLAG);
        textPaint.setTextSize(48);
        textPaint.setColor(Color.BLUE);
        textPaint.setTextAlign(Paint.Align.CENTER);
        canvas.drawColor(Color.WHITE);
        //(1)
        BlurMaskFilter blur =
                    new BlurMaskFilter(100, BlurMaskFilter.Blur.NORMAL);
        paint.setMaskFilter(blur);
        canvas.drawCircle(300, 240, 150, paint);
        canvas.drawText("NORMAL", 300, 460, textPaint);
```



项目：CanvasDrawTest
maskfilter_view

```
        paint.setMaskFilter(new BlurMaskFilter(100, INNER)); //(2)
        canvas.drawCircle(800, 240, 150, paint);
        canvas.drawText("INNER", 800, 460, textPaint);
         //(3)
        paint.setMaskFilter(new BlurMaskFilter(100, OUTER));
        canvas.drawCircle(300, 660, 150, paint);
        canvas.drawText("OUTER", 300, 880, textPaint);
        //(4)
        paint.setMaskFilter(new BlurMaskFilter(100, SOLID));
        canvas.drawCircle(800, 660, 150, paint);
        canvas.drawText("SOLID", 800, 880, textPaint);
        //(5)
        paint.setColor(Color.BLUE);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(20);
        paint.setTextSize(180); //文字大小（px）
        EmbossMaskFilter emboss =  //光源方向 环境光亮度 镜面反射系数 模糊半径
                new EmbossMaskFilter(new float[]{10, 10, 10}, 0.1f, 5f, 5f);
        paint.setMaskFilter(emboss);
        canvas.drawText("中山大学", 220, 1100, paint);
        //(6)
        paint.setStyle(Paint.Style.FILL);
        paint.setTextSize(120);
        paint.setStrokeWidth(20);
        canvas.drawText("计算机学院", 280, 1280, paint);
    }
}
```



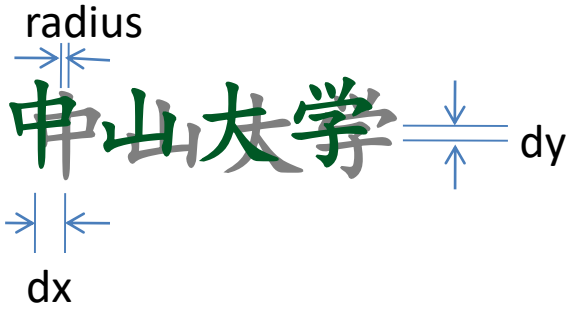
设置阴影层
(setShadowLayer)

参考1

- 通过为画笔设置阴影层，可以为形状、文字增加阴影，但是不能为图像增加阴影。如果需要图像增加阴影，需要先做一副alpha图，然后把原图叠加上去。

```
Paint.setShadowLayer(float radius,float dx,float dy,int color)
```

radius	*阴影半径*
dx	*阴影在X轴的位移*
dy	*阴影在Y轴的位移*
color	*阴影颜色*



例：

```
paint.setShadowLayer(10, 12, 8, Color.DKGRAY);
```

- 对于形状，例如，画圆，需要以下语句取消硬件加速后才能显示出阴影。

```
setLayerType(LAYER_TYPE_SOFTWARE, null);
```

```java
public class ShadowView extends View {
    Context context;
    public ShadowView(Context context, AttributeSet attrs) {
        super(context, attrs);  this.context = context;
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        //取消硬件加速（*）
        setLayerType(LAYER_TYPE_SOFTWARE, null);
        Paint paint = new Paint();
        //（1）
        paint.setColor(Color.RED);
        paint.setStrokeWidth(10);
        paint.setStyle(Paint.Style.STROKE);
        paint.setShadowLayer(10, 12, 8, Color.DKGRAY);
        canvas.drawCircle(460, 120, 100, paint);
         //（2）
        paint.reset();
        paint.setColor(Color.rgb(0,88,37));
        // 阴影半径, X 轴位移, Y 轴位移, 阴影颜色
        paint.setShadowLayer(10, 10, 2, Color.DKGRAY);
        paint.setTextSize(160);
        canvas.drawText("中山大学", 200, 420, paint);
```

blog12.png

项目：CanvasDrawTest
shadow_view

```
        canvas.translate(160, 480);
        paint.reset();
        Bitmap bitmap = BitmapFactory.decodeResource(
                  getResources(), R.drawable.blog12);
        int width = 600;
        int height = width*bitmap.getHeight()
                                 /bitmap.getWidth();
        Bitmap alphaBmp = bitmap.extractAlpha();

         //绘制阴影(3)
        paint.setColor(Color.DKGRAY);
        paint.setMaskFilter(new BlurMaskFilter(10,
                        BlurMaskFilter.Blur.NORMAL));
        canvas.drawBitmap(alphaBmp, null,
              new Rect(10, 10, width, height), paint);

        //绘制阴影 （4）
        canvas.translate(0, 400);
        canvas.drawBitmap(alphaBmp, null,
         new Rect(10, 10, width, height), paint);
        //绘制原图像（4）
        paint.setMaskFilter(null);
        canvas.drawBitmap(bitmap, null,
                           new Rect(0, 0, width, height), paint);
    }
}
```

项目：CanvasDrawTest
shadow_view

设置颜色过滤器（setColorFilter）

- 画笔的这个方法采用ColorFilter实例作为参数，实际上是使用其子类ColorMatrixColorFilter、LightingColorFilter或PorterDuffColorFilter的实例作为参数。

- 类ColorMatrixColorFilter的构造器采用一个4×5的矩阵作为参数：

```
Public ColorMatrixColorFilter(ColorMatrix matrix)
```

颜色过滤器利用该矩阵对图像的每个像素点进行如下颜色变换：

$$\begin{pmatrix} R' \\ G' \\ B' \\ A' \end{pmatrix} = \begin{pmatrix} a & b & c & d & e \\ f & g & h & i & j \\ k & l & m & n & o \\ p & q & r & s & t \end{pmatrix} \times \begin{pmatrix} R \\ G \\ B \\ A \\ 1 \end{pmatrix} = \begin{pmatrix} a*R+b*G+c*B+d*A+e \\ f*R+g*G+h*B+i*A+j \\ k*R+l*G+m*B+n*A+o \\ p*R+q*G+r*B+s*A+t \end{pmatrix}$$

现颜色　　ColorMatrix　　原颜色

```
colorMatrix = new ColorMatrix(new float[]{
        1/4f, 1/4f, 1/4f, 0, 0,
        1/4f, 1/4f, 1/4f, 0, 0,
        1/4f, 1/4f, 1/4f, 0, 0,
        0,       0   , 0, 1, 0
});
paint.setColorFilter(new ColorMatrixColorFilter(colorMatrix));
canvas.drawBitmap(bitmap, 0, 0, paint);
```

原图

转换后的图

如果
```
colorMatrix = new ColorMatrix(new float[]{
            1/2f, 1/2f, 1/2f, 0,  0,
            1/4f, 1/4f, 1/4f, 0,  0,
            1/4f, 1/4f, 1/4f, 0,  0,
            0,       0   , 0, 1,  0
    });
```

?

- 类LightingColorFilter的构造器采用一个3或者4字节的乘数mul和一个3或4个字节的加数add（3个字节只包含RGB）：

```
Public LightingColorFilter(int mul, int add)
```

颜色过滤器对图像的每个像素点先用mul相乘（与操作），然后加上add的值进行如下颜色变换

```
LightingColorFilter light=
                    new LightingColorFilter(0xFFFF0000, 0x00000000);
paint.setColorFilter(light);
canvas.drawBitmap(bitmap, 0, 0, paint);
```

?

mul=0xFFFF0000
add=0x000000FF

?

CanvasDrawTest
color_filter_view

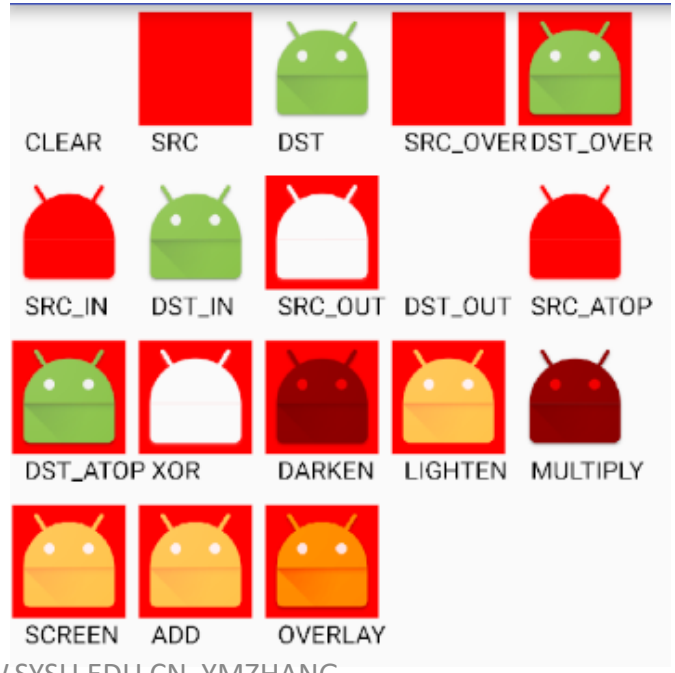
- 类PorterDuffColorFilter的构造器采用一个颜色值和一个叠加模式实现一种叠加过滤：

```
public void PorterDuffColorFilter(@ColorInt int color,
                                  @NonNull PorterDuff.Mode mode)
```

颜色过滤器先用颜色值得到源图（单颜色矩形），然后再把目标图用某种方式叠加到源图上。

```
paint.setColorFilter(new PorterDuffColorFilter(Color.RED,
                                               PorterDuff.Mode.DARKEN));
canvas.drawBitmap(bitmap, null, new Rect(xOffset, yOffset,
                        xOffset + width, yOffset + height), paint);
```

这是下面程序的运行结果：



项目：CanvasDrawTest
porterduff_view

```
public class PorterDuffView extends View {
    Context context;
    int xOffset, yOffset, xDelta, yDelta, canvasWidth;
    public PorterDuffView(Context context, AttributeSet attrs) {
        super(context, attrs);  this.context = context;
    }
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvasWidth=canvas.getWidth();
        xOffset = 10; yOffset = 10;
        xDelta = 200; yDelta = 260;
        Paint paint = new Paint();
        paint.setColor(Color.BLACK); paint.setTextSize(40);
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
                                                     R.mipmap.ic_launcher);
        int width = 180;
        int height = width * bitmap.getHeight() / bitmap.getWidth();
        for(PorterDuff.Mode mode:PorterDuff.Mode.class.getEnumConstants()){
            paint.setColorFilter(new PorterDuffColorFilter(Color.RED, mode));
            canvas.drawBitmap(bitmap, null, new Rect(xOffset, yOffset,
                                  xOffset + width, yOffset + height), paint);
            paint.setColorFilter(null);
            canvas.drawText(mode.toString(),xOffset+20,yOffset+height+40,paint);
            updatePos();
        }
    }
```

```
    void updatePos() {
        if(xOffset + xDelta+100>= canvasWidth){
            xOffset=10;
            yOffset+=yDelta;
        }
        else{
            xOffset +=xDelta;
        }
    }
}
```

ColorMatrix的主要方法：

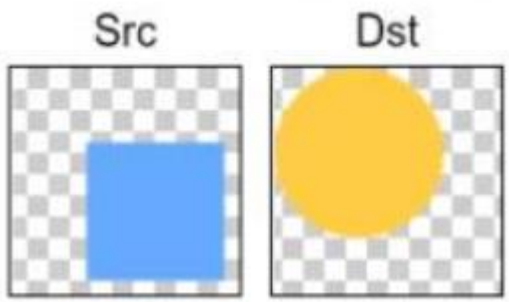
set(float[] src);	*设置颜色矩阵数组*
set(ColorMatrix src);	*设置颜色矩阵*
setConcat(ColorMatrix a, ColorMatrix b);	*连结ab矩阵；效果为先应用b后应用a*
setSaturation(float sat);	*设置色彩的饱和度*
setScale(float rScale, float gScale, float bScale, float aScale);	*设置用于缩放(即乘法)的RGBA值，原始比例为1*
setRotate(int axis, float degrees);	*绕axis轴旋转degrees度；axis：0为RED，1为GREEN，2为BLUE*

PorterDuff

参考

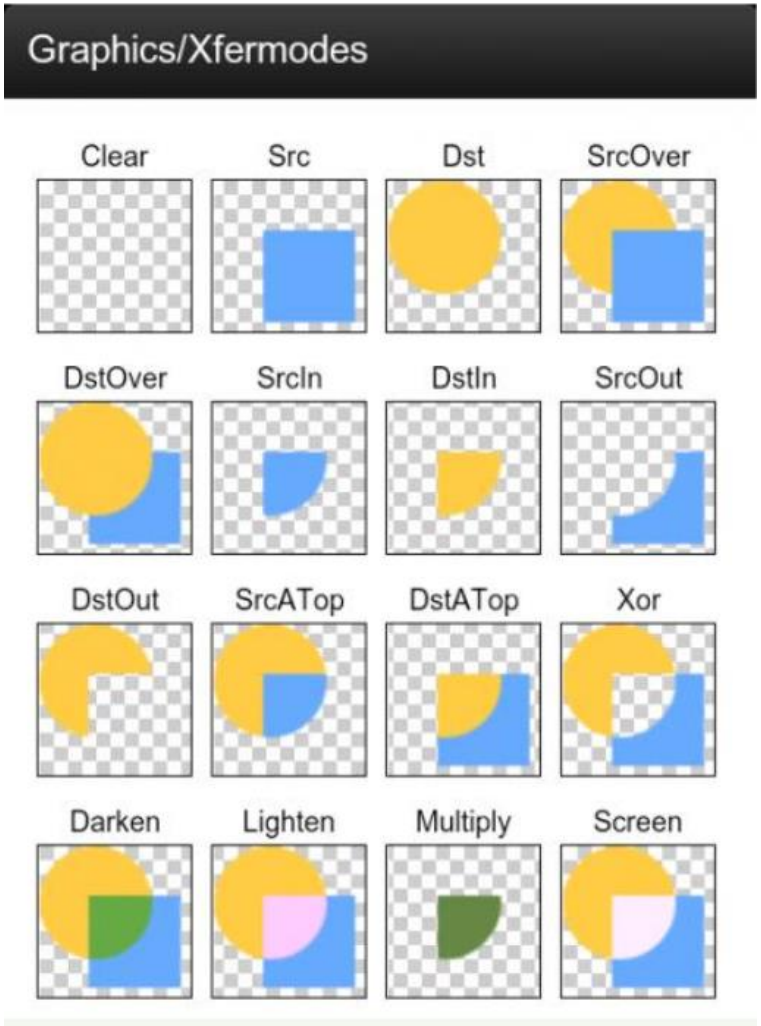
- 概述
  - PorterDuff是由Tomas Proter和Tom Duff提出的图形混合的方法。先绘制的图是目标图(DST)，后绘制的图是源图(SRC)，DST和SRC的绘制方法共有18种混合模式。

```
canvas.drawBitmap(dstBitmap, xOffset, yOffset, paint);
PorterDuff.Mode mode = PorterDuff.Mode.DST_OVER;
PorterDuffXfermode porterDuffXfermode
                        = new PorterDuffXfermode(mode);
paint.setXfermode(porterDuffXfermode);
canvas.drawBitmap(srcBitmap, xOffset, yOffset, paint);
```



CLEAR 不显示源图
SRC 只保留源图像
DST 只保留目标图
SRC_OVER 把源图绘制在上方
DST_OVER 目标图绘制在上方
SRC_IN 在相交处绘制源图
DST_IN 在相交处绘制目标图
SRC_OUT 不相交处绘制源图
DST_OUT 不相交处绘制目标图

SRC_ATOP 相交处绘制源图，不相交处绘制目标图
DST_ATOP 相交处绘制目标图，不相交处绘制源图
XOR 不相交的地方按原样绘制源图和目标图
DARKEN 取两图的全部区域，交集部分颜色加深
LIGHTEN 取两图的全部区域，点亮交集部分颜色
MULTIPLY 取两图层交集部分叠加后颜色
SCREEN 取两图层全部区域，交集部分变为透明色
ADD 饱和度叠加
OVERLAY 叠加



- PorterDuff.Mode

参考1 参考2

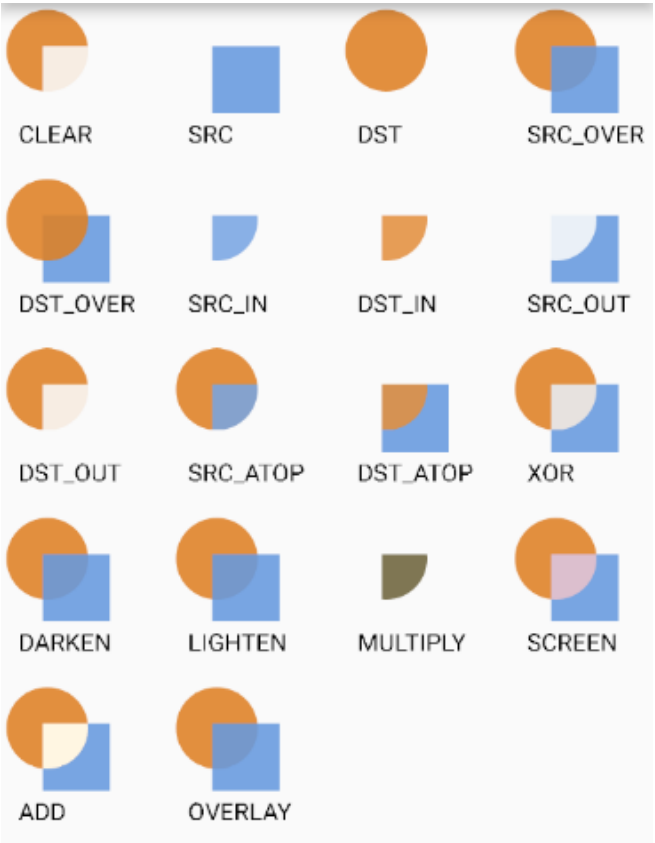
1	ADD	饱和度叠加	Saturate(S + D)
2	CLEAR	把原图	[0, 0]
3	DARKEN	取两图层全部区域，交集部分颜色加深	[Sa + Da - Sa*Da, Sc*(1 - Da) + Dc*(1 - Sa) + min(Sc, Dc)]
4	DST	只保留目标图	[Da, Dc]
5	DST_ATOP	源图和目标图相交处绘制目标图，不相交的地方绘制源图	[Sa, Sa * Dc + Sc * (1 - Da)]
6	DST_IN	两者相交的地方绘制目标图，绘制的效果会受到原图处的透明度影响	[Sa * Da, Sa * Dc]
7	DST_OUT	在不相交的地方绘制目标图	[Da * (1 - Sa), Dc * (1 - Sa)]
8	DST_OVER	目标图绘制在上方	[Sa + (1 - Sa)*Da, Rc = Dc + (1 - Da)*Sc]
9	LIGHTEN	取两图层全部区域，点亮交集部分颜色	[Sa + Da - Sa*Da, Sc*(1 - Da) + Dc*(1 - Sa) + max(Sc, Dc)]
10	MULTIPLY	取两图层交集部分叠加后颜色	[Sa * Da, Sc * Dc]
11	OVERLAY	叠加	
12	SCREEN	取两图层全部区域，交集部分变为透明色	[Sa + Da - Sa * Da, Sc + Dc - Sc * Dc]
13	SRC	只保留源图像	[Sa, Sc]
14	SRC_ATOP	源图和目标图相交处绘制源图，不相交的地方绘制目标图	[Da, Sc * Da + (1 - Sa) * Dc]
15	SRC_IN	两者相交的地方绘制源图	[Sa * Da, Sc * Da]
16	SRC_OUT	不相交的地方绘制源图	[Sa * (1 - Da), Sc * (1 - Da)]
17	SRC_OVER	把源图绘制在上方	[Sa + (1 - Sa)*Da, Rc = Sc + (1 - Sa)*Dc]
18	XOR	不相交的地方按原样绘制源图和目标图	[Sa + Da - 2 * Sa * Da, Sc * (1 - Da) + (1 - Sa) * Dc]

* D指原本在Canvas上的内容dst，S指绘制输入的内容src，a指alpha通道，c指RGB各个通道。结果图：[alpha,color]

项目：CanvasDrawTest
porterduff_xfer_view

```
public class PorterDuffXferView extends View {
    Context context;
    int xOffset, yOffset, xDelta, yDelta;
    int screenW,screenH;
    Bitmap srcBitmap,dstBitmap;
    //源图和目标图宽高
    private int width = 180;
    private int height = 180;
    public PorterDuffXferView(Context context, AttributeSet attrs) {
        super(context, attrs);  this.context = context;
        screenW = ScreenUtil.getScreenW(context);
        screenH = ScreenUtil.getScreenH(context);
        srcBitmap = makeSrc(width, height);
        dstBitmap = makeDst(width, height);
    }
    //创建一个圆形bitmap，作为dst图
    private Bitmap makeDst(int w, int h) {
        Bitmap bm = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
        Canvas c = new Canvas(bm);
        Paint p = new Paint(Paint.ANTI_ALIAS_FLAG);
        p.setColor(0xDFDF8024);//255 160 68
        c.drawOval(new RectF(0, 0, w * 3 / 4, h * 3 / 4), p);
        return bm;
    }
```

```
private Bitmap makeSrc(int w, int h) {
    Bitmap bm = Bitmap.createBitmap(w, h,
                    Bitmap.Config.ARGB_8888);
    Canvas c = new Canvas(bm);
    Paint p = new Paint(Paint.ANTI_ALIAS_FLAG);
    p.setColor(0xDF669ADF);
    c.drawRect(new RectF(w/3, h/3,
              w*19/20, h*19/20), p);
    return bm;
}
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    xOffset = 10; yOffset = 10;
    xDelta = width+100; yDelta = height+100;
    Paint textPaint= new Paint();
    textPaint.setColor(Color.BLACK);
    textPaint.setTextSize(40);

    for(PorterDuff.Mode mode:PorterDuff.Mode.class.getEnumConstants()){
        drawXfer(canvas,mode);
        canvas.drawText(mode.toString(),xOffset+20,yOffset+height+40,
                        textPaint);
        updatePos();
    }
}
```



```
    void drawXfer(Canvas canvas,PorterDuff.Mode mode) {
        Paint paint = new Paint();
        int sc = canvas.saveLayer(0, 0, screenW, screenH, null,
                 Canvas.MATRIX_SAVE_FLAG |
                Canvas.CLIP_SAVE_FLAG |
                Canvas.HAS_ALPHA_LAYER_SAVE_FLAG |
                Canvas.FULL_COLOR_LAYER_SAVE_FLAG |
                Canvas.CLIP_TO_LAYER_SAVE_FLAG);
        paint.setXfermode(null);
        canvas.drawBitmap(dstBitmap, xOffset, yOffset, paint);
        PorterDuffXfermode porterDuffXfermode
                  = new PorterDuffXfermode(mode);
        paint.setXfermode(porterDuffXfermode);
        canvas.drawBitmap(srcBitmap, xOffset,
                                  yOffset, paint);
        canvas.restoreToCount(sc);
    }
    void updatePos() {
        if(xOffset + xDelta+100>= screenW){
            xOffset=10;
            yOffset+=yDelta;
        }
        else{
            xOffset +=xDelta;
        }
    }
}
```

如果makeDst和makeSrc用以下函数代替：

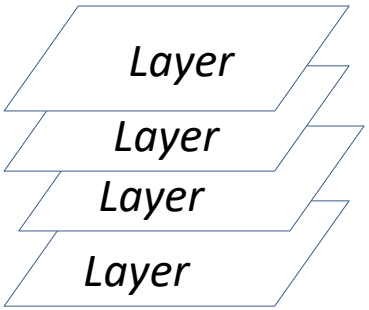
```
private Bitmap makeDst(int w, int h) {
    Bitmap bm = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
    Canvas c = new Canvas(bm); Paint p = new
Paint(Paint.ANTI_ALIAS_FLAG);
    p.setColor(0xDFDF8024);
    c.drawRect(new RectF(0, 0, w * 3 / 4, h * 3 / 4), p);
    return bm;
}
private Bitmap makeSrc(int w, int h) {
    Bitmap bm = Bitmap.createBitmap(w,h,Bitmap.Config.ARGB_8888);
    Canvas c = new Canvas(bm);
    Paint p = new Paint(Paint.ANTI_ALIAS_FLAG);

    Bitmap bitmap=BitmapFactory.decodeResource(
            getResources(), R.mipmap.ic_launcher);
    c.drawBitmap(bitmap,null,
           new RectF(w/3,h/3,w*19/20,h*19/20),p);
    return bm;
}
```

得到的结果：

saveLayer

参考1 参考2

- Canvas默认是单层的，所有的绘图操作如drawBitmap, drawCircle都发生在这层上进行。
- 如果要进行复杂的绘图操作，比如绘制包含政区层、道路层等多个层次的地图，就需要使用Canvas的多图层（Layer）功能。
- Canvas使用saveLayer和restore创建多个中间层，并按照栈结构进行管理：



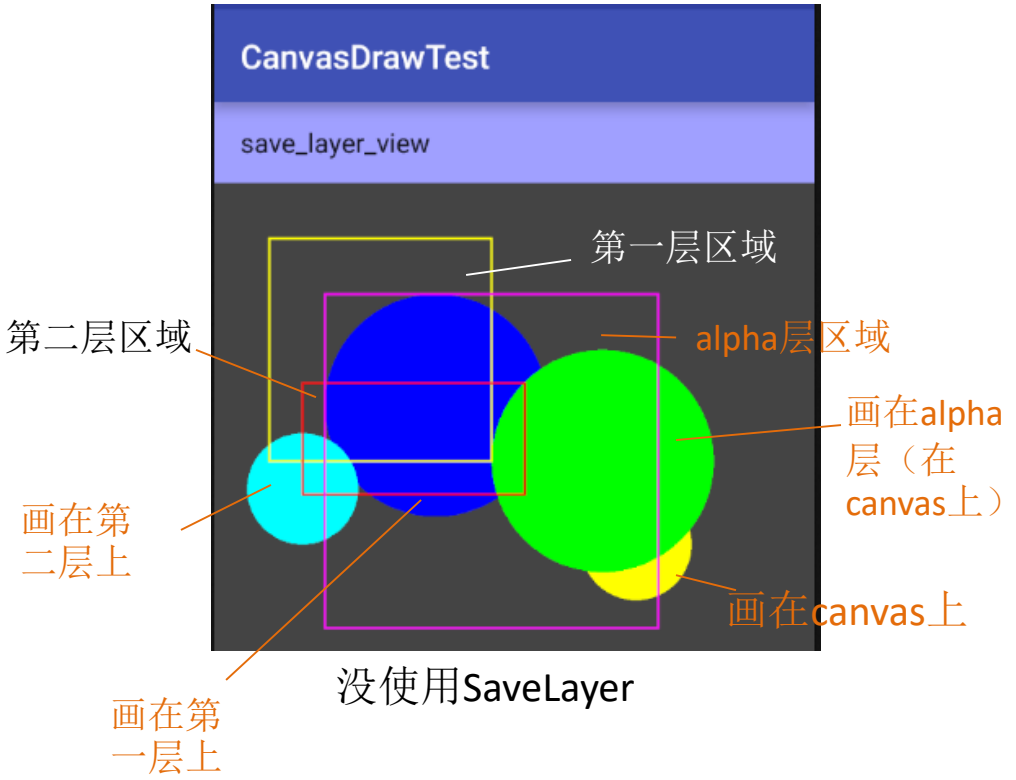
- 每次使用saveLayer时就会创建一个新的Layer进栈，后续的绘图操作都发生在这个Layer上，使用restore时会从栈中退出一个Layer，该Layer上绘制的图像都会被绘制到栈中的上层Layer上。没有上层Layer时才被绘制到Canvas上。使用restoreToCount可以一次使多个Layer退栈。
- Canvas的saveLayerAlpha为透明层，所有在透明层上绘制的图像都是半透明的。
- Paint的方法setShadowLayer产生一个用于绘制阴影的Shadow层，使得绘制的形状和文字都会出现阴影。

public int saveLayer(RectF bounds, Paint paint, int saveFlags)

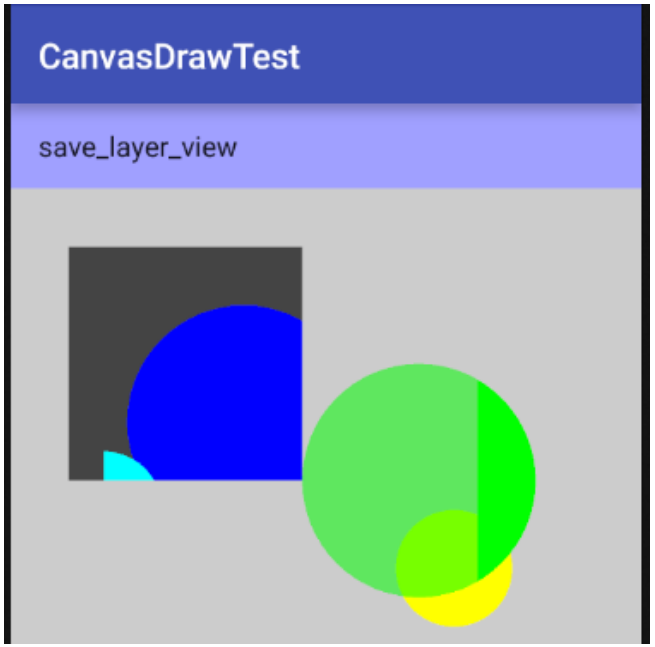
saveFlags表示要保存哪方面的内容，这里一共有6种取值：

MATRIX_SAVE_FLAG
HAS_ALPHA_LAYER_SAVE_FLAG
CLIP_TO_LAYER_SAVE_FLAG
CLIP_SAVE_FLAG
FULL_COLOR_LAYER_SAVE_FLAG
ALL_SAVE_FLAG

下页的例子的运行结果：



没使用SaveLayer



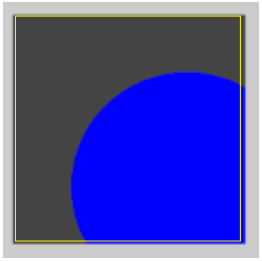
使用SaveLayer

Canvas层

```
Paint paint = new Paint();
canvas.drawColor(Color.LTGRAY);
```

第一层

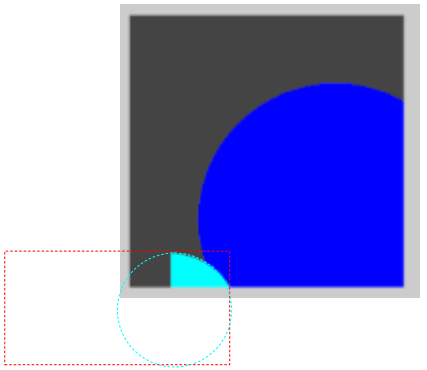
```
int saveCount = canvas.saveLayer(100, 100, 500, 500,
                                 null, Canvas.ALL_SAVE_FLAG);
canvas.drawColor(Color.DKGRAY);
paint.setColor(Color.BLUE);
canvas.drawCircle(400, 400, 200, paint);
canvas.restore();
```



加入

第二层

```
canvas.saveLayer(160, 360, 560, 560,
                 null, Canvas.ALL_SAVE_FLAG);
paint.setColor(Color.CYAN);
canvas.drawCircle(160, 550, 100, paint);
canvas.restore();
```

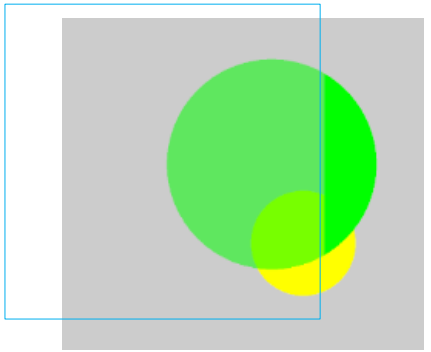


Canvas层

```
paint.setColor(Color.YELLOW);
canvas.drawCircle(760, 650, 100, paint);
```

第一层

```
canvas.saveLayerAlpha(200,200,800,800,
     0x88,Canvas.HAS_ALPHA_LAYER_SAVE_FLAG);
paint.setColor(Color.GREEN);
canvas.drawCircle(700, 500, 200, paint);
canvas.restore();
```



项目名：CanvasDrawTest
save_layer_view

```
public class SaveLayerView extends View {
    public SaveLayerView(Context context, AttributeSet attrs){super(context,attrs);}
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();
        canvas.drawColor(Color.LTGRAY);
        int saveCount=canvas.saveLayer(100,100,500,500,null,Canvas.ALL_SAVE_FLAG);
        canvas.drawColor(Color.DKGRAY);
        paint.setColor(Color.BLUE);
        canvas.drawCircle(400, 400, 200, paint);
        canvas.saveLayer(160, 360, 560, 560, null, Canvas.ALL_SAVE_FLAG);
        paint.setColor(Color.CYAN);
        canvas.drawCircle(160, 550, 100, paint);
        canvas.restore();
        canvas.restore();
        //canvas.restoreToCount(2); //可以替换上面两个语句
        paint.setColor(Color.YELLOW);
        canvas.drawCircle(760, 650, 100, paint);
        canvas.saveLayerAlpha(200,200,800,800,0x88,Canvas.HAS_ALPHA_LAYER_SAVE_FLAG);
        paint.setColor(Color.GREEN);
        canvas.drawCircle(700, 500, 200, paint);
        canvas.restore();
    }
}
```

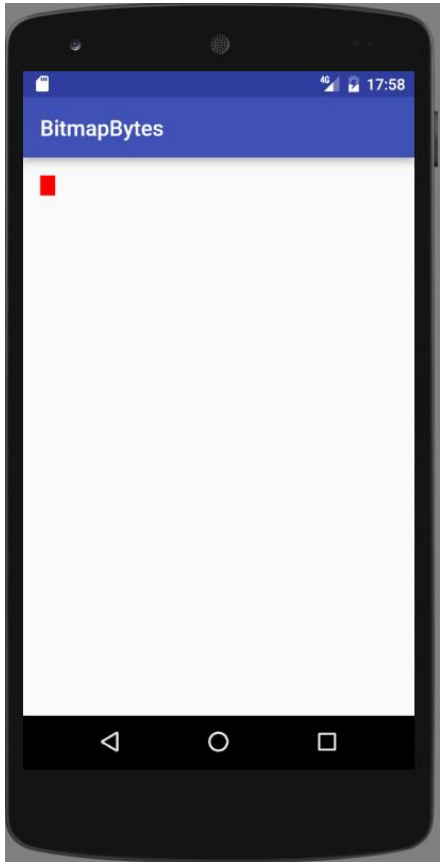
附录

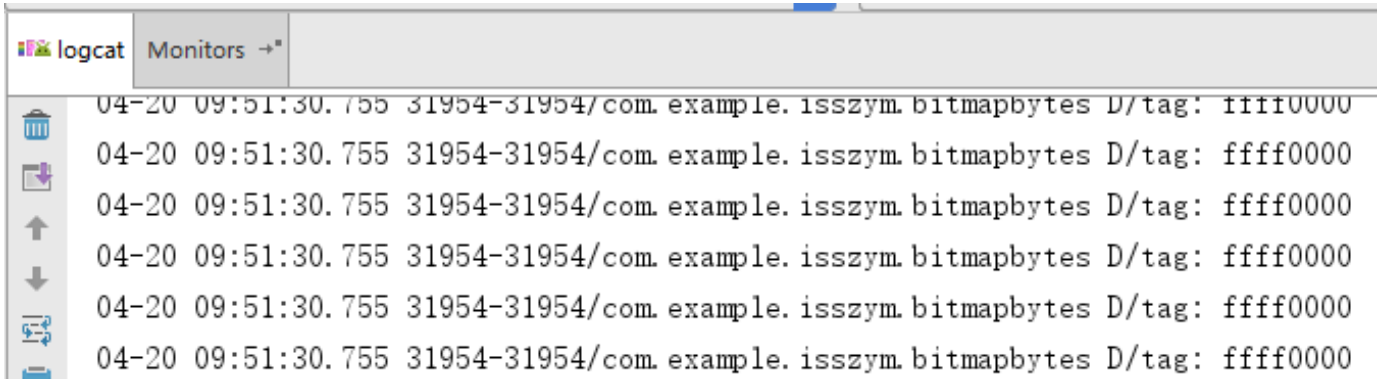
附录1、取控件(View)像素

项目：BitmapBytes

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        View tv=findViewById(R.id.textView);
        getPicturePixel(getBitmapFromView(tv));
    }
    public Bitmap getBitmapFromView(View view) {
        view.destroyDrawingCache();
        view.measure(View.MeasureSpec.makeMeasureSpec(0,View.MeasureSpec.UNSPECIFIED),
                View.MeasureSpec.makeMeasureSpec(0, View.MeasureSpec.UNSPECIFIED));
        view.layout(0, 0, view.getMeasuredWidth(), view.getMeasuredHeight());
        view.setDrawingCacheEnabled(true);
        Bitmap bitmap = view.getDrawingCache(true);
        return bitmap;
    }
    private void getPicturePixel(Bitmap bitmap) {
        int width = bitmap.getWidth();
        int height = bitmap.getHeight();
        // 保存所有的像素的数组，图片宽×高
        int[] pixels = new int[width * height];
        bitmap.getPixels(pixels, 0, width, 0, 0, width, height);
        for (int i = 0; i < pixels.length; i++) {
            int clr = pixels[i];    // int red = (clr & 0x00ff0000) >> 16; // 取高两位
            Log.d("tag",String.format("%08x", clr));
        }
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/activity_vertical_margin"
    tools:context="com.example.isszym.bitmapbytes.MainActivity">
    android:id="@+id/textView"

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="  "
        android:textColor="#FFFFFFFF"
        android:background="#FFFF0000" />
</RelativeLayout>
```





附录2、Bitmap.createBitmap函数

1、public static Bitmap createBitmap (int width, int height, Bitmap.Config config)
根据参数创建新位图

2、public static Bitmap createBitmap (Bitmap src)
从原位图src复制出一个新的位图，和原始位图相同

3、public static Bitmap createBitmap (Bitmap src, int x, int y, int width, int height)
从原位图src复制一部分形成新位图。x为起始x坐标，y为起始y坐标，width为要截的图的宽度，height为要截的图的宽度。

4、public static Bitmap createBitmap (Bitmap src, int x, int y, int width, int height, Matrix m, boolean filter)
从原位图src复制一部分再用m进行变换后形成新位图。

5、public static Bitmap createBitmap (int[] colors, int width, int height, Bitmap.Config config)
根据颜色数组创建位图，数组长度>=width*height。先用width和height创建空位图，然后用颜色数组colors来从左到右从上至下一次填充颜色。

6、public static Bitmap createBitmap (int[] colors, int offset, int stride, int width, int height, Bitmap.Config config)
此方法与2类似，offset为偏移量，stride为步幅，即从数组的offset开始每隔stride取数组元素填充位图颜色。

附录3、Bitmap.Config 详解

- ALPHA_8：每个像素信息只存储了alpha（透明度）这一项信息。
- ARGB_4444： 这个值在level 13的时候就已经不被建议使用了，因为他是一个低质量的配置，它被建议使用 ARGB_8888。
- ARGB_8888：每个像素信息占用4个字节（即32个二进制位）的存储空间，alpha、red、green、blue各占8个二进制位。这个配置项是非常灵活，并提供了最好的质量。
- RGB_565：每个像素信息占用2个字节（即16位二进制位）只存储了RGB的信息没有alpha信息，其中Red5位，Green6位，Blue5位。这个配置项在不需要提供透明度的情况下更有用。

附录4、Bitmap对象与字节数组

1、将Bitmap对象读到字节数组中

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
bitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos);
byte[] datas = baos.toByteArray();
```

*compress的第二个参数为图片质量（压缩率），100表示最好质量。

2、将字节数组转为Bitmap对象

```
byte[] b = getIntent().getByteArrayExtra("bitmap");
Bitmap bitmap = BitmapFactory.decodeByteArray(b, 0, b.length);
```

附录5、画笔的属性 参考

reset()	*paint恢复为默认设置*
set(src)	*从src目录中装载一种画笔的样式*
setAlpha(a)	设置画笔的透明度
setARGB(a, r, g, b)	设置画笔的ARGB颜色色值
setColor(color)	设置画笔的颜色
setColorFilter(filter)	设置画笔的过滤器
setDither(dither)	设置清除抖动对图像的影响，会使绘制出来的图片图像更加清晰，颜色更加平滑和饱满
setFakeBoldText(fakeBoldText)	设置粗体文字，注意设置在小字体上效果会非常差
setFilterBitmap(filter)	如果该项设置为true，则图像在动画进行中会滤掉对Bitmap图像的优化操作，加快速度显示。 注意:本设置项依赖于dither和xfermode的设置
setFlags(flags)	给画笔打上标记，以便直接复用
setHinting(mode)	设置画笔的隐藏模式

setLinearText(linearText)	设置线性文本
setMaskFilter(maskfilter)	设置MaskFilter，可以用不同的MaskFilter实现滤镜的效果，如滤化，立体等
setPathEffect(effect)	设置绘制路径的效果，如点画线等
setRasterizer(rasterizer)	设置或清除的光栅对象。 通过空清除任何以前的光栅化。为方便起见，传递的参数也回来了。
setShader(shader)	设置或清除渲染对象。 通过空清以前的任何材质。为方便起见，传递的参数也回来了。
setShadowLayer(radius, dx, dy, color)	这原图层下面设置阴影层，产生阴影效果，radius为阴影的角度，dx和dy为阴影在x轴和y轴上的距离，color为阴影的颜色。如果半径为0，然后一层阴影去除。
setStrikeThruText(strikeThruText)	设置画笔带有删除线效果的
setStrokeCap(cap)	当画笔样式为STROKE或FILL_OR_STROKE时，设置笔刷的图形样式，如圆形样式 Cap.ROUND或方形样式Cap.SQUARE
setStrokeMiter(miter)	设置笔画的倾斜度（无效果）
setStrokeJoin(join)	设置画笔结合处的样式，Miter:结合处为锐角， Round:结合处为圆弧：BEVEL：结合处为直线。
setStrokeWidth(width)	当画笔是空心样式时，设置画笔空心的宽度
setStyle(style)	Paint.Style.FILL(实心) Paint.Style.STROKE(空心) Paint.Style.FILL_AND_STROKE(实心带边)

setSubpixelText(subpixelText)	设置该项为true，将有助于文本在LCD屏幕上的显示效果
setTextAlign(align)	设置绘制文字的对齐方向
etTextLocale(locale)	设置文本的区域比如中文、日文等
setTextScaleX(scaleX)	设置文本的水平方向的缩放比例
setTextSize(textSize)	设置文本的字体大小
setTextSkewX(skewX)	设置斜体文字，skewX为倾斜弧度
setTypeface(typeface)	设置Typeface对象，即字体风格，包括粗体，斜体以及衬线体，非衬线体等
setUnderlineText(underlineText)	设置带有下划线的文字效果
setXfermode(xfermode)	设置图形重叠时的处理方式，如合并，取交集或并集，经常用来制作橡皮的擦除效果

附录6、获取屏幕尺寸

```
public class ScreenUtil {
    /** 获取屏幕宽高，sdk17后不建议采用 */
    public static int[] getScreenHW(Context context) {
        WindowManager manager = (WindowManager) context.getSystemService(Context.WINDOW_SERVICE);
        Display display = manager.getDefaultDisplay();
        int width = display.getWidth();
        int height = display.getHeight();
        int[] HW = new int[] { width, height };
        return HW;
    }
    /** 获取屏幕宽高，建议采用  */
    public static int[] getScreenHW2(Context context) {
        WindowManager manager = (WindowManager) context.getSystemService(Context.WINDOW_SERVICE);
        DisplayMetrics dm = new DisplayMetrics();
        manager.getDefaultDisplay().getMetrics(dm);
        int width = dm.widthPixels;
        int height = dm.heightPixels;
        int[] HW = new int[] { width, height };
        return HW;
    }
    public static int getScreenW(Context context) { return getScreenHW2(context)[0]; }
    public static int getScreenH(Context context) { return getScreenHW2(context)[1]; }
}
```

附录7、Matrix的其他方法

参考

public Matrix() // 构造器，使用单位矩阵
public Matrix(Matrix src) // 构造器
public boolean isIdentity() // 判断是否是单位矩阵
public boolean isAffine() // 判断是否是仿射矩阵
public boolean rectStaysRect() // 判断该矩阵是否可以将一个矩形依然变换为一个矩形。public void reset() // 重置为单位矩阵。
public boolean setRectToRect(RectF src, RectF dst, ScaleToFit stf)

将rect变换成rect， ScaleToFit 有如下四个值：

FILL: 可能会变换矩形的长宽比，保证变换和目标矩阵长宽一致。

START:保持坐标变换前矩形的长宽比，并最大限度的填充变换后的矩形。至少有一边和目标矩形重叠。左上对齐。

CENTER: 保持坐标变换前矩形的长宽比，并最大限度的填充变换后的矩形。至少有一边和目标矩形重叠。

END:保持坐标变换前矩形的长宽比，并最大限度的填充变换后的矩形。至少有一边和目标矩形重叠。右下对齐。

public boolean setPolyToPoly(float[] src, int srcIndex,float[] dst, int dstIndex,int pointCount)

通过指定的0-4个点，原始坐标以及变化后的坐标，来得到一个变换矩阵。如果指定0个点则没有效果。

public boolean invert(Matrix inverse)

反转当前矩阵，如果能反转就返回true并将反转后的值写入inverse，否则返回false。当前矩阵*inverse=单位矩阵。

public void mapPoints(float[] dst, int dstIndex, float[] src, int srcIndex,int pointCount) public void mapPoints(float[] dst, float[] src)

public void mapPoints(float[] pts)

映射点的值到指定的数组中，这个方法可以在矩阵变换以后，给出指定点的值。

dst：指定写入的数组

dstIndex：写入的起始索引，x，y两个坐标算作一对，索引的单位是对，也就是经过两个值才加1

src：指定要计算的点

srcIndex：要计算的点的索引

pointCount：需要计算的点的个数，每个点有两个值，x和y。

public void mapVectors(float[] dst, int dstIndex, float[] src, int srcIndex,int vectorCount) public void mapVectors(float[] dst, float[] src)

public void mapVectors(float[] vecs)

与上面的mapPoionts基本类似，这里是将一个矩阵作用于一个向量，由于向量的平移前后是相等的，所以这个方法不会对translate相关的方法产生反应，如果只是调用了translate相关的方法，那么得到的值和原本的一致。

public boolean mapRect(RectF dst, RectF src)

public boolean mapRect(RectF rect)

返回值即是调用的rectStaysRect()，这个方法前面有讲过，这里把src中指定的矩形的左上角和右下角的两个点的坐标，写入dst中。

public float mapRadius(float radius)

返回一个圆圈半径的平均值，将matrix作用于一个指定radius半径的圆，随后返回的平均半径。