Bitmap

android.graphics.Bitmap

```
public final class Bitmap implements Parcelable

public static Bitmap createBitmap(@NonNull Bitmap src)

public static Bitmap createBitmap(int width, int height, @NonNull Config config)

public static Bitmap createBitmap(@NonNull Bitmap source,
        int x, int y, int width, int height)

public static Bitmap createBitmap(@NonNull Bitmap source,
        int x, int y, int width, int height, @Nullable Matrix m, boolean filter)

public static Bitmap createScaledBitmap(@NonNull Bitmap src,
        int dstWidth, int dstHeight, boolean filter)

public final boolean isMutable() //返回true，才能把Canvas中的图形输出到Bitmap对象中

public final boolean isRecycled() //true表示已回收。已回收的Bitmap对象不能再绘制到Canvas中

public boolean compress(CompressFormat format, int quality, OutputStream stream)
    //将Bitmap转换成输出流，此方法可以保存一个Bitmap对象到本地
    //参数format指定图片压缩格式：jpg、png、
    //参数quality指定图片压缩质量，范围0-100，0表示最低质量，100表示最高质量。png格式会忽略质量
    //已回收的Bitmap对象无法被压缩

public enum Config {...}
```

android.graphics.Bitmap.CompressFormat

```
public enum CompressFormat

JPEG    (0),
PNG     (1),
WEBP    (2);
```

android.graphics.Bitmap.Config

```
ALPHA_8     (1),

RGB_565     (3),

@Deprecated
ARGB_4444   (4),

ARGB_8888   (5),
```

android.graphics.BitmapFactory

```
public class BitmapFactory

public static Bitmap decodeFile(String pathName)
public static Bitmap decodeFile(String pathName, Options opts)

public static Bitmap decodeResource(Resources res, int id)
public static Bitmap decodeResource(Resources res, int id, Options opts)
public static Bitmap decodeStream(InputStream is)
public static Bitmap decodeStream(InputStream is, Rect outPadding, Options opts)

public static Bitmap decodeByteArray(byte[] data, int offset, int length)
public static Bitmap decodeByteArray(byte[] data, int offset, int length, Options opts)

public static Bitmap decodeFileDescriptor(FileDescriptor fd)
public static Bitmap decodeFileDescriptor(FileDescriptor fd,
        Rect outPadding, Options opts)


public static class Options {...}
```

android.graphics.BitmapFactory.Options

```
public Options()

public boolean inJustDecodeBounds;

public int inSampleSize;
```

Drawable

android.graphics.drawable.Drawable

```
public abstract class Drawable

public int getIntrinsicWidth() //Drawable的固有宽度

public int getIntrinsicHeight() //Drawable的固有高度

public abstract void draw(@NonNull Canvas canvas);
        //每个Drawable的子类都需要重写这个draw方法，用于将Drawable表示的图形绘制到Canvas画布中

public abstract void setAlpha(@IntRange(from=0,to=255) int alpha);
public int getAlpha()

public abstract @PixelFormat.Opacity int getOpacity();

public abstract void setColorFilter(@Nullable ColorFilter colorFilter);

public void setBounds(@NonNull Rect bounds)
        //指定将Drawable对象的Rect矩形范围内的图形绘制到Canvas画布中
        //此方法必须在调draw方法之前执行，否则Drawable表示的图形无法绘制在Canvas画布上。
```

android.graphics.drawable.ColorDrawable

```
public class ColorDrawable extends Drawable

public ColorDrawable(@ColorInt int color) //将颜色值转成一个Drawable对象
```

android.graphics.drawable.BitmapDrawable

```
public class BitmapDrawable extends Drawable

public BitmapDrawable(Resources res, Bitmap bitmap)

public BitmapDrawable(Resources res, String filepath)

public BitmapDrawable(Resources res, java.io.InputStream is)
```

```
@attr ref android.R.styleable#BitmapDrawable_src
@attr ref android.R.styleable#BitmapDrawable_antialias
@attr ref android.R.styleable#BitmapDrawable_filter
@attr ref android.R.styleable#BitmapDrawable_dither
@attr ref android.R.styleable#BitmapDrawable_gravity
@attr ref android.R.styleable#BitmapDrawable_mipMap
@attr ref android.R.styleable#BitmapDrawable_tileMode

public void setAntiAlias(boolean aa)
public boolean hasAntiAlias()

public void setFilterBitmap(boolean filter)
public boolean isFilterBitmap()

public void setDither(boolean dither)

public void setGravity(int gravity)

public void setMipMap(boolean mipMap)

public void setTileModeX(Shader.TileMode mode)
public final void setTileModeY(Shader.TileMode mode)
public void setTileModeXY(Shader.TileMode xmode, Shader.TileMode ymode)
    //Shader.TileMode.CLAMP|REPEAT|MIRROR
```

android.graphics.drawable.GradientDrawable （根标签为`<shape>`的图片）

```
public class GradientDrawable extends Drawable

<shape>标签的属性
android.R.styleable#GradientDrawable_visible
android.R.styleable#GradientDrawable_shape
android.R.styleable#GradientDrawable_innerRadiusRatio
android.R.styleable#GradientDrawable_innerRadius
android.R.styleable#GradientDrawable_thicknessRatio
android.R.styleable#GradientDrawable_thickness
android.R.styleable#GradientDrawable_useLevel

<size>标签的属性
android.R.styleable#GradientDrawableSize_width
android.R.styleable#GradientDrawableSize_height

<gradient>标签的属性
android.R.styleable#GradientDrawableGradient_startColor
android.R.styleable#GradientDrawableGradient_centerColor
android.R.styleable#GradientDrawableGradient_endColor
android.R.styleable#GradientDrawableGradient_useLevel
android.R.styleable#GradientDrawableGradient_angle
android.R.styleable#GradientDrawableGradient_type
android.R.styleable#GradientDrawableGradient_centerX
android.R.styleable#GradientDrawableGradient_centerY
android.R.styleable#GradientDrawableGradient_gradientRadius

<solid>标签的属性
android.R.styleable#GradientDrawableSolid_color

<stroke>标签的属性
android.R.styleable#GradientDrawableStroke_width
android.R.styleable#GradientDrawableStroke_color
android.R.styleable#GradientDrawableStroke_dashWidth
android.R.styleable#GradientDrawableStroke_dashGap

<padding>标签的属性
android.R.styleable#GradientDrawablePadding_left
android.R.styleable#GradientDrawablePadding_top
android.R.styleable#GradientDrawablePadding_right
android.R.styleable#GradientDrawablePadding_bottom

<corners>标签的属性
android.R.styleable#DrawableCorners_radius
android.R.styleable#DrawableCorners_topLeftRadius
android.R.styleable#DrawableCorners_topRightRadius
android.R.styleable#DrawableCorners_bottomLeftRadius
android.R.styleable#DrawableCorners_bottomRightRadius
```

android.graphics.drawable.DrawableContainer

```
public class DrawableContainer extends Drawable implements Drawable.Callback
```

android.graphics.drawable.StateListDrawable (根标签为`<selector>`的图片)

```
public class StateListDrawable extends DrawableContainer

@attr ref android.R.styleable#StateListDrawable_visible
@attr ref android.R.styleable#StateListDrawable_variablePadding
@attr ref android.R.styleable#StateListDrawable_constantSize
@attr ref android.R.styleable#StateListDrawable_dither

@attr ref android.R.styleable#StateListDrawableItem_drawable

@attr ref android.R.styleable#DrawableStates_state_focused
@attr ref android.R.styleable#DrawableStates_state_window_focused
@attr ref android.R.styleable#DrawableStates_state_enabled
@attr ref android.R.styleable#DrawableStates_state_checkable
@attr ref android.R.styleable#DrawableStates_state_checked
@attr ref android.R.styleable#DrawableStates_state_selected
@attr ref android.R.styleable#DrawableStates_state_activated
@attr ref android.R.styleable#DrawableStates_state_active
@attr ref android.R.styleable#DrawableStates_state_single
@attr ref android.R.styleable#DrawableStates_state_first
@attr ref android.R.styleable#DrawableStates_state_middle
@attr ref android.R.styleable#DrawableStates_state_last
@attr ref android.R.styleable#DrawableStates_state_pressed
```

android.graphics.drawable.LayerDrawable (根标签为`<layer-list>`的图片)

```
public class LayerDrawable extends Drawable implements Drawable.Callback

@attr ref android.R.styleable#LayerDrawable_paddingMode

@attr ref android.R.styleable#LayerDrawableItem_left
@attr ref android.R.styleable#LayerDrawableItem_top
@attr ref android.R.styleable#LayerDrawableItem_right
@attr ref android.R.styleable#LayerDrawableItem_bottom
@attr ref android.R.styleable#LayerDrawableItem_start
@attr ref android.R.styleable#LayerDrawableItem_end
@attr ref android.R.styleable#LayerDrawableItem_width
@attr ref android.R.styleable#LayerDrawableItem_height
@attr ref android.R.styleable#LayerDrawableItem_gravity
@attr ref android.R.styleable#LayerDrawableItem_drawable
@attr ref android.R.styleable#LayerDrawableItem_id
```

绘图

android.graphics.BaseCanvas

```
@hide
public abstract class BaseCanvas
```

android.graphics.Canvas

```
public class Canvas extends BaseCanvas

public Canvas()

public Canvas(@NonNull Bitmap bitmap)
    //bitmap Specifies a mutable bitmap for the canvas to draw into

public void setBitmap(@Nullable Bitmap bitmap)
    //bitmap Specifies a mutable bitmap for the canvas to draw into

public void drawBitmap(@NonNull Bitmap bitmap,
        float left, float top, @Nullable Paint paint)
    //在Canvas中left、top参数指定的位置，将源图bitmap用指定的画笔paint绘制到Canvas中

public void drawBitmap(@NonNull Bitmap bitmap,
        @Nullable Rect src, @NonNull RectF dst, @Nullable Paint paint)

public void drawBitmap(@NonNull Bitmap bitmap,
        @Nullable Rect src, @NonNull Rect dst, @Nullable Paint paint)
    //从源图biamp中挖取src范围的图片，用指定的画笔paint绘制在Canvas指定的dst范围中。

public void drawBitmap(@NonNull Bitmap bitmap,
        @NonNull Matrix matrix, @Nullable Paint paint)

public void drawArc(@NonNull RectF oval, float startAngle, float sweepAngle,
        boolean useCenter, @NonNull Paint paint)
    //绘制的弧线是矩形内切椭圆oval的部分边线，以3点钟方向为0度角，顺时针绘制，起始点为startAngle
    //指定的椭圆上的某点，结束点为sweepAngle指定的椭圆上的某点。
    //startAngle为负数或大于等于360时，取startAngele%360
    //sweepAngle为负数时，取sweepAngle%360；大于等于360时，表示绘制整个椭圆

public void drawArc(float left, float top, float right, float bottom,
        float startAngle, float sweepAngle, boolean useCenter, @NonNull Paint paint)
    //left、top、right、bottom参数确定Canvas上的四个顶点，以这四个顶点做矩形的内切椭圆

public void drawOval(@NonNull RectF oval, @NonNull Paint paint)

public void drawOval(float left, float top, float right, float bottom,
        @NonNull Paint paint)

public void drawCircle(float cx, float cy, float radius, @NonNull Paint paint)
    //cx，cy指定圆心在Canvas上的坐标（cx,cy），radius表示半径

public void drawLine(float startX, float startY, float stopX, float stopY,
        @NonNull Paint paint)
    //以Canvas内坐标（startX,startY）为起点，坐标（stopX,stopY）为终点，
    //使用画笔paint绘制一条直线

public void drawLines(@Size(multiple = 4) @NonNull float[] pts, int offset, int count,
        @NonNull Paint paint)
    //按照索引顺序，pts数组中每4个元素作为起点和终点坐标绘制一条直线，需要绘制的直线数量为count/4
    //offset表示数组中绘制第一条直线的起始x坐标，count表示一共有多少个数组元素参与绘制直线
    //如果count数量不足4个，那么一条直线也绘制不出来；如果count数量不是4的整数倍，那么能绘制的
    //直线数量就是 count>>2（即count/4），即使有剩余的数组元素，也绘制不出直线

public void drawLines(@Size(multiple = 4) @NonNull float[] pts,
        @NonNull Paint paint)
    //pts数组中的元素表示 [startx0,starty0,endx0,endy0,startx1,starty1,endx1,endy1,...]

public void drawPoint(float x, float y, @NonNull Paint paint)
    //以Canvas内的坐标(x,y)绘制一个点

public void drawPoints(@Size(multiple = 2) float[] pts, int offset, int count,
        @NonNull Paint paint)
    //参数意思跟drawLines类似，只不过drawPoints是通过pts数组确定多个坐标，来绘制多个点

public void drawPoints(@Size(multiple = 2) @NonNull float[] pts,
        @NonNull Paint paint)

public void drawRect(@NonNull RectF rect, @NonNull Paint paint)
    //以RectF确定的要绘制的直角矩形范围，根据Paint对象确定是只画矩形边框，还是用颜色填充矩形

public void drawRect(@NonNull Rect r, @NonNull Paint paint)
    //以Rect确定的要绘制的直角矩形范围

public void drawRect(float left, float top, float right, float bottom,
        @NonNull Paint paint)
    //以left|top|right|bottom 确定的要绘制的直角矩形范围

public void drawRoundRect(@NonNull RectF rect, float rx, float ry,
        @NonNull Paint paint)
    //绘制圆角矩形，RectF确定矩形范围，
    //rx,ry分别表示圆角在x轴和y轴上的半径。rx,ry不相等时可绘制椭圆圆角

public void drawRoundRect(float left, float top, float right, float bottom,
        float rx, float ry, @NonNull Paint paint)

public void drawPath(@NonNull Path path, @NonNull Paint paint)
    //沿着Path指定的形状进行绘制

public void drawText(@NonNull char[] text, int index, int count,
        float x, float y, @NonNull Paint paint)
    //根据index，count从char[]数组中取出要绘制的字符串
    //坐标(x,y)表示由Paint对象的对齐方式决定的绘制字符串的起始位置

public void drawText(@NonNull String text,
        float x, float y, @NonNull Paint paint)

public void drawText(@NonNull String text, int start, int end,
        float x, float y, @NonNull Paint paint)
    //根据start，end从String对象text中取出要绘制的字符串

public void drawText(@NonNull CharSequence text, int start, int end,
        float x, float y, @NonNull Paint paint)
    //根据start，end从CharSequence对象text中取出要绘制的字符串

public void drawTextOnPath(@NonNull char[] text, int index, int count,
    @NonNull Path path, float hOffset, float vOffset, @NonNull Paint paint)
    //沿着Path指定的图形的路径绘制字符串
    //hOffset 表示在水平方向上，偏移图形路径起始位置的hOffset处放置字符串
    //vOffset 表示在竖直方向上，偏移图形路径的vOffset处放置字符串。
    //负值表示向上偏移，正值表示向下偏移

public void drawTextOnPath(@NonNull String text,
    @NonNull Path path, float hOffset, float vOffset, @NonNull Paint paint)

public boolean clipRect(@NonNull RectF rect)   //将Canvas中RectF矩形区域裁剪掉
public boolean clipOutRect(@NonNull RectF rect) //将Canvas中RectF矩形之外的区域裁剪掉

public boolean clipRect(@NonNull Rect rect)
public boolean clipOutRect(@NonNull Rect rect)

public boolean clipRect(float left, float top, float right, float bottom)
public boolean clipOutRect(float left, float top, float right, float bottom)
public boolean clipRect(int left, int top, int right, int bottom)
public boolean clipOutRect(int left, int top, int right, int bottom)

public void translate(float dx, float dy)
    //移动Canvas。dx为正表示向右移动，为负向左移动；dy为正向下移动，为负向上移动

public void scale(float sx, float sy)
public final void scale(float sx, float sy, float px, float py) {
    if (sx == 1.0f && sy == 1.0f) return;
    translate(px, py);
    scale(sx, sy);
    translate(-px, -py);
}
    //以轴点（px,py）对Canvas执行缩放变换

public void rotate(float degrees)
public final void rotate(float degrees, float px, float py) {
    if (degrees == 0.0f) return;
    translate(px, py);
    rotate(degrees);
    translate(-px, -py);
}
    //以轴点（px,py）对Canvas执行旋转变换

public void skew(float sx, float sy) //对Canvas执行倾斜变换
```

android.graphics.Paint

```
public class Paint

public Paint()

public Paint(Paint paint) //使用参数paint提供的画笔属性创建一个新的Paint对象

public void setAlpha(int a)  //设置画笔颜色的透明度，取值范围[0,255]

public void setAntiAlias(boolean aa)
    //是否开启抗锯齿功能。开启后，图形的边界会比较平滑，对图形内部无影响

public final boolean isAntiAlias() {
    return (getFlags() & ANTI_ALIAS_FLAG) != 0;
}

public void setColor(@ColorInt int color) //设置画笔颜色

public void setARGB(int a, int r, int g, int b) { //设置画笔颜色
    setColor((a << 24) | (r << 16) | (g << 8) | b);
}

public PathEffect setPathEffect(PathEffect effect)
    //设置绘制路径时的路径效果。如果参数为null，表示清除之前的路径效果
    //直接返回参数对象

public Shader setShader(Shader shader)
    //设置画笔的填充效果，Shader表示着色器，填充效果就是指用什么颜色来填充
    //如果参数为null，表示清除之前的着色器

public void setShadowLayer(float radius, float dx, float dy, int shadowColor)
    //在main layer下绘制一个阴影图层

public void clearShadowLayer() {
    setShadowLayer(0, 0, 0, 0);
}
```

```
public void setStrokeWidth(float width) //设置画笔笔触的宽度,
    //Pass 0 to stroke in hairline mode
    //Hairlines always draws a single pixel independent of the canva's matrix

public void setStrokeJoin(Paint.Join join) //设置画笔笔触转弯处的连接风格

public void setStyle(Paint.Style style) //设置画笔的填充风格,
    //填充风格包括只将图形内部区域用颜色填充，还是只将图形边界用颜色描边,
    //还是图形内部区域和图形边界都进行颜色填充
    //默认是Styly.FILL 只将图形内部区域用颜色填充

public void setTextAlign(Align align)
    //设置drawText绘制字符串时，字符串在绘制起点处的对齐方式
    //默认是Align.LEFT，即向左靠着绘制起点进行绘制，向左靠着起点就是说字符串在起点位置的右边

public void setTextSize(float textSize)  //设置绘制字符串的大小，以px为单位，必须大于0

public float measureText(String text)
public float measureText(String text, int start, int end)
    //测量字符串的宽度

public ColorFilter setColorFilter(ColorFilter filter)
```

android.graphics.Paint.Join

```
public enum Join

MITER   (0), //The outer edges of a join meet at a sharp angle

ROUND   (1),  //The outer edges of a join meet in a circular arc.

BEVEL   (2);  //The outer edges of a join meet with a straight line
```

android.graphics.Paint.Style

```
public enum Style

FILL    (0),

STROKE  (1),

FILL_AND_STROKE (2);
```

android.graphics.Paint.Align

```
public enum Align

LEFT    (0),  //The text is drawn to the right of the x,y origin

CENTER  (1),  //The text is drawn centered horizontally on the x,y origin

RIGHT   (2);  //The text is drawn to the left of the x,y origin
```

android.graphics.ColorMatrixColorFilter

```
public class ColorMatrixColorFilter extends ColorFilter

public ColorMatrixColorFilter(@NonNull ColorMatrix matrix)
```

android.graphics.PorterDuffColorFilter

```
public class PorterDuffColorFilter extends ColorFilter

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

android.graphics.ColorMatrix

```
public class ColorMatrix

public ColorMatrix()

public void set(float[] src) //20个元素组成4x5的矩阵

public void setScale(float rScale, float gScale, float bScale, float aScale)

public void setRotate(int axis, float degrees)

public void setSaturation(float sat)

public void setConcat(ColorMatrix matA, ColorMatrix matB)

public void preConcat(ColorMatrix prematrix) {
    setConcat(this, prematrix);
}

public void postConcat(ColorMatrix postmatrix) {
    setConcat(postmatrix, this);
}

public void setRGB2YUV()

public void setYUV2RGB()
```

android.graphics.Path

```
public class Path

public Path()

public Path(Path src)

public void reset()
    //清除Path中所有的直线和曲线

public void rewind()
    //清除Path中所有的直线和曲线，但保留此Path对象设置的数据，以便下次重新使用

public void moveTo(float x, float y)
    //从坐标点(x,y)开始绘制一条路径

public void rMoveTo(float dx, float dy)
    //将上一次调moveTo方法确定的起点坐标(x,y)，再加上偏移量dx和dy,
    //重新确定一条路径的起始点坐标(x+dx, y+dy)
    //如果之前没有调用moveTo方法，那么执行此方法，相当于执行了moveTo(0+dx, 0+dy)

public void lineTo(float x, float y)
    //将参数坐标点(x,y)和上一次的路径坐标点连成一条直线
    //如果之前没有执行moveTo方法确定起始坐标，那么以坐标原点(0,0)作为起始点

public void rLineTo(float dx, float dy)
    //将上一次的路径坐标点(x,y)，再加上偏移量dx,dy，确定一个新的坐标点(x+dx, y+dy)，并将上一次
        路径坐标点(x,y) 和新的坐标点(x+dx, y+dy)连成一条直线
    //如果不存在上次的路径坐标点，那么会先执行moveTo(0,0)确定一个起始点，再将坐标点(0, 0)和
        坐标点(0+dx, 0+dy)连成一条直线
```

```
public void quadTo(float x1, float y1, float x2, float y2)
    //从上一次路径坐标点(x0,y0)开始，以(x1,y1)为曲线控制坐标点，(x2,y2)为曲线结束点，绘制一条
        二次的贝塞尔曲线
    //如果不存在上次的路径坐标点，则以(0,0)为起始点

public void rQuadTo(float dx1, float dy1, float dx2, float dy2)
    //从上一次路径坐标点(x0,y0)开始，以(x0+dx1, y0+dy1)为曲线控制坐标点，(x0+dx2, y0+dy2)
        为曲线结束点，绘制一条二次的贝塞尔曲线

public void cubicTo(float x1, float y1, float x2, float y2, float x3, float y3)
    //从上一次路径坐标点(x0,y0)开始，以(x1,y1)为第一曲线控制坐标点，(x2,y2)为第二曲线控制点，
        (x3,y3)为曲线结束点，绘制一条三次的贝塞尔曲线

public void rCubicTo(float x1, float y1, float x2, float y2, float x3, float y3)

public void close()
    //结束当前图形路径，如果结束时的当前坐标点不是图形路径的起始点，那么会用一条直线
        将结束时的当前坐标点和路径的起始点连接起来。所以，绘制曲线图时，最后不能调close方法

public void arcTo(RectF oval, float startAngle, float sweepAngle, boolean forceMoveTo)
    //在Path中指定一个弧形路径
    //如果forceMoveTo为false，并且此Path对象之前有绘制过图形，那么会将弧形的绘制起点和
        当前坐标点（即上次绘制图形的最后坐标点）连接起来
    //如果forceMoveTo为true,或者此Path对象之前没有绘制过图形，那么会将弧形的绘制起点作为此Path
        对象的起始绘制坐标点

public void arcTo(RectF oval, float startAngle, float sweepAngle) //forceMoveTo为false

public void arcTo(float left, float top, float right, float bottom,
        float startAngle, float sweepAngle, boolean forceMoveTo)

*****************************************************

public void addRect(RectF rect, Direction dir)
public void addRect(float left, float top, float right, float bottom, Direction dir)

public void addRoundRect(RectF rect, float rx, float ry, Direction dir)
public void addRoundRect(float left, float top, float right, float bottom,
        float rx, float ry, Direction dir)

public void addOval(RectF oval, Direction dir)
public void addOval(float left, float top, float right, float bottom, Direction dir)

public void addCircle(float x, float y, float radius, Direction dir)

public void addArc(RectF oval, float startAngle, float sweepAngle)
public void addArc(float left, float top, float right, float bottom,
        float startAngle, float sweepAngle)

public void addPath(Path src, float dx, float dy)
public void addPath(Path src)
public void addPath(Path src, Matrix matrix)

public void offset(float dx, float dy)
public void offset(float dx, float dy, @Nullable Path dst)

public void setLastPoint(float dx, float dy)

public void transform(Matrix matrix)
public void transform(Matrix matrix, Path dst)
```

android.graphics.Path.Direction

```
public enum Direction
```

```
CW  (0),  // clockwise

CCW (1);  // counter-clockwise
```

android.graphics.CornerPathEffect

```
public class CornerPathEffect extends PathEffect

public CornerPathEffect(float radius)
    //如果Path对象中的图形路径中存在尖锐的顶角，绘制时会用半径为radius的圆角替换掉这个尖锐的顶角
    //默认情况下，两直线段相交处都为尖锐的顶角
    //此PathEffect对象可用于绘制平滑的曲线
```

android.graphics.DiscretePathEffect

```
public class DiscretePathEffect extends PathEffect

public DiscretePathEffect(float segmentLength, float deviation)
    //对于Path对象中的图形路径，将其分割成segmentLength长度的多条线段，每条线段以deviation指定
        的偏差，偏移原图形路径方向进行绘制
    //当线段足够短时，配合适宜的偏移量，就能得到一条“带刺”的图形路径，
    //可用于绘制一条有干扰信号的曲线
```

android.graphics.DashPathEffect

```
public class DashPathEffect extends PathEffect

public DashPathEffect(float intervals[], float phase)
    //将原Path对象中连续的图形路径，用虚线绘制出来。（虚线就是有间隔的破折线）
    //虚线是由破折线、空白间隔、破折线、空白间隔、...组成。每个破折线和空白间隔的长度都可设置，
        于是数组参数intervals的作用是：按照索引顺序，偶数索引元素定义破折线长度，奇数索引元素
            定义空白间隔长度。
        并且数组的长度要≥2，并且是2的整数倍。如果是长度＞2但为奇数的话，最后一个元素虽然是定义
            破折线长度的，但不会被绘制出来。
    //绘制虚线路径时，将intervals数组看作一组虚线，重复绘制
    //phase表示绘制的起始偏移量，偏移量可以无限增大，但由于每组虚线是重复出现的，所以当偏移量
        达到一组虚线的总长度时（总长度就是intervals数组元素之和），显示效果上偏移量又回到了0。
    //在onDraw方法中调用invalid()方法重复绘制时，不断改变phase偏移量，则Path图形路径会产生
        动态显示的效果。
    //当paint's style 为STROKE或STROKE_AND_FILL时DashPathEffect才起作用
```

android.graphics.PathDashPathEffect

```
public class PathDashPathEffect extends PathEffect

public PathDashPathEffect(Path shape, float advance, float phase, Style style)
    //使用Path对象shape绘制的图形作为stamp印记代替破折线显示，
        advance表示两个stamp印记之间的间隔，advance的值必须大于印记图形shape的宽度
        才能显示出间隔，其实 空白间隔 = advance - shape.width
```

android.graphics.PathDashPathEffect.Style

```
public enum Style

TRANSLATE  (0),   //!< translate the shape to each position
ROTATE     (1),   //!< rotate the shape about its center
MORPH       (2);   //!< transform each point, and turn lines into curves
```

android.graphics.ComposePathEffect

```
public class ComposePathEffect extends PathEffect

public ComposePathEffect(PathEffect outerpe, PathEffect innerpe)
    //通过两个不同的PathEffect对象，构造出一个新的PathEffect对象
    //以inner PathEffect的效果为基础，然后再包装上out PathEffect的效果进行显示
    // outer(inner(path))
```

android.graphics.SumPathEffect

```
public class SumPathEffect extends PathEffect

public SumPathEffect(PathEffect first, PathEffect second)
    //通过将两个不同的PathEffect对象的显示效果依次叠加，形成一个新的PathEffect对象
    //first(path) + second(path)
```