



河北师范大学软件学院
Software College of Hebei Normal University



Android 基础开发

第七章 第一讲 Android中图形的处理



Java与移动智能设备开发



教学目标

- 掌握Android中简单图形的绘制



目录

- 1 Android中简单的图片使用
- 2 Android中的绘图
- 3 Android中的图形特效



Android中简单的图片使用

- Android中的图片基本分为两种：
 - 应用程序中的系统图片：一般在Mipmap或Drawable文件夹中。
 - 引入程序外部图片：一般在内存中，或者SDCard中（后续课程中讲解）。



Android中简单的图片使用

- 一般的图片的用途：






Android中Drawable图片使用


- 当在Android应用增加图片对象后，会在R类中找到它对应的ID，通过如下两种方式可以访问：
 - 在XML布局文件中
 - @mipmap/filename
 - @drawable/filename
 - 在Java代码中
 - R.mipmap.filename
 - R.drawable.filename




Android中外部图片使用


- Android应用中的外部图片，一般会被表示成一个Bitmap对象。
- Bitmap类提供了一系列的静态方法来得到对象：


 `createBitmap(Bitmap src) : Bitmap - Bitmap`

 `createBitmap(int width, int height, Config config) : Bitmap - Bitmap`

 `createBitmap(int[] colors, int width, int height, Config config) : Bitmap - Bitmap`

 `createBitmap(Bitmap source, int x, int y, int width, int height) : Bitmap - Bitmap`

 `createBitmap(int[] colors, int offset, int stride, int width, int height, Config config) : Bitmap - Bitmap`

 `createBitmap(Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter) : Bitmap - Bitmap`



Android中外部图片使用

- BitmapFactory提供了一系列的方法从外部数据源得到Bitmap对象。

```
• decodeByteArray(byte[] data, int offset, int length) : Bitmap - BitmapFactory
• decodeByteArray(byte[] data, int offset, int length, Options opts) : Bitmap - BitmapFactory
• decodeFile(String pathName) : Bitmap - BitmapFactory
• decodeFile(String pathName, Options opts) : Bitmap - BitmapFactory
• decodeFileDescriptor(FileDescriptor fd) : Bitmap - BitmapFactory
• decodeFileDescriptor(FileDescriptor fd, Rect outPadding, Options opts) : Bitmap - BitmapFactory
• decodeResource(Resources res, int id) : Bitmap - BitmapFactory
• decodeResource(Resources res, int id, Options opts) : Bitmap - BitmapFactory
• decodeResourceStream(Resources res, TypedValue value, InputStream is, Rect pad, Options opts) : Bitmap - BitmapFactory
• decodeStream(InputStream is) : Bitmap - BitmapFactory
• decodeStream(InputStream is, Rect outPadding, Options opts) : Bitmap - BitmapFactory
```




Bitmap与Drawable转换

- 将资源文件转化成 Bitmap

```
Resources res = getResources();  
Bitmap bmp = BitmapFactory.decodeResource(res,  
    R.mipmap.ic_drawable);
```

- 将资源文件转化成 Drawable

```
Drawable drawable =  
getResources().getDrawable(R.mipmap.ic);
```

- Bitmap 转换成 Drawable

```
Drawable drawable = new BitmapDrawable(bmp);
```



课堂练习

- <http://blog.csdn.net/gdliweibing/article/details/43195327>
- <http://blog.csdn.net/gdliweibing/article/details/43206283>



目录

- 1 Android中简单的图片使用
- 2 **Android中的绘图**
- 3 Android中的图形特效



重要的类



- Bitmap：相当于画布。
- Paint：画笔。
- Canvas：相当于图形的模具。
- Path：绘制路径。
- Shader：填充图形的渲染类。



Android中的绘图

- Android中经常需要在运行时动态生成图片。
- Android中使用Canvas类来实现：
 - 其中还涉及到的类有Paint类，Path类等。

```
• drawArc(RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint) : void - Canvas
• drawARGB(int a, int r, int g, int b) : void - Canvas
• drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint) : void - Canvas
• drawBitmap(Bitmap bitmap, float left, float top, Paint paint) : void - Canvas
• drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint) : void - Canvas
• drawBitmap(Bitmap bitmap, Rect src, RectF dst, Paint paint) : void - Canvas
• drawBitmap(int[] colors, int offset, int stride, float x, float y, int width, int height, boolean hasAlpha, Paint paint) : void - Canvas
• drawBitmap(int[] colors, int offset, int stride, int x, int y, int width, int height, boolean hasAlpha, Paint paint) : void - Canvas
• drawBitmapMesh(Bitmap bitmap, int meshWidth, int meshHeight, float[] verts, int vertOffset, int[] colors, int colorOffset, Paint p
• drawCircle(float cx, float cy, float radius, Paint paint) : void - Canvas
• drawColor(int color) : void - Canvas
• drawColor(int color, Mode mode) : void - Canvas
• drawLine(float startX, float startY, float stopX, float stopY, Paint paint) : void - Canvas
• drawLines(float[] pts, Paint paint) : void - Canvas
• drawLines(float[] pts, int offset, int count, Paint paint) : void - Canvas
• drawOval(RectF oval, Paint paint) : void - Canvas
• drawPaint(Paint paint) : void - Canvas
```



Android中的绘图

- Canvas可以理解为画布，绘画的工具称为画笔Paint类。
- Paint类提供了画笔的常见属性设置。

```
● set(Paint src) : void - Paint
● setAlpha(int a) : void - Paint
● setAntiAlias(boolean aa) : void - Paint
● setARGB(int a, int r, int g, int b) : void - Paint
● setColor(int color) : void - Paint
● setColorFilter(ColorFilter filter) : ColorFilter - Paint
● setDither(boolean dither) : void - Paint
● setFakeBoldText(boolean fakeBoldText) : void - Paint
● setFilterBitmap(boolean filter) : void - Paint
● setFlags(int flags) : void - Paint
● setLinearText(boolean linearText) : void - Paint
● setMaskFilter(MaskFilter maskfilter) : MaskFilter - Paint
● setPathEffect(PathEffect effect) : PathEffect - Paint
● setRasterizer(Rasterizer rasterizer) : Rasterizer - Paint
● setShader(Shader shader) : Shader - Paint
● setShadowLayer(float radius, float dx, float dy, int color) : void - Paint
● setStrikeThruText(boolean strikeThruText) : void - Paint
```



Android中的绘图

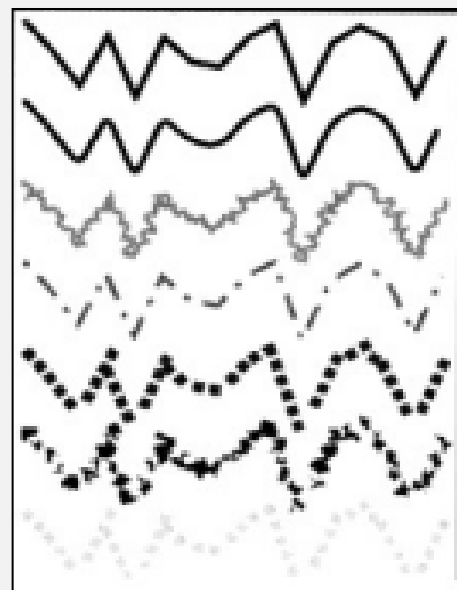
- Canvas可以理解为画布，画布上的形状都是由线组成。
- Path代表任意多条直线连接成的图形。

```
● addArc(RectF oval, float startAngle, float sweepAngle) : void - Path
● addCircle(float x, float y, float radius, Direction dir) : void - Path
● addOval(RectF oval, Direction dir) : void - Path
● addPath(Path src) : void - Path
● addPath(Path src, Matrix matrix) : void - Path
● addPath(Path src, float dx, float dy) : void - Path
● addRect(RectF rect, Direction dir) : void - Path
● addRect(float left, float top, float right, float bottom, Direction dir) : void - Path
● addRoundRect(RectF rect, float[] radii, Direction dir) : void - Path
● addRoundRect(RectF rect, float rx, float ry, Direction dir) : void - Path
● arcTo(RectF oval, float startAngle, float sweepAngle) : void - Path
● arcTo(RectF oval, float startAngle, float sweepAngle, boolean forceMoveTo) : void - Path
```



Android中的绘图

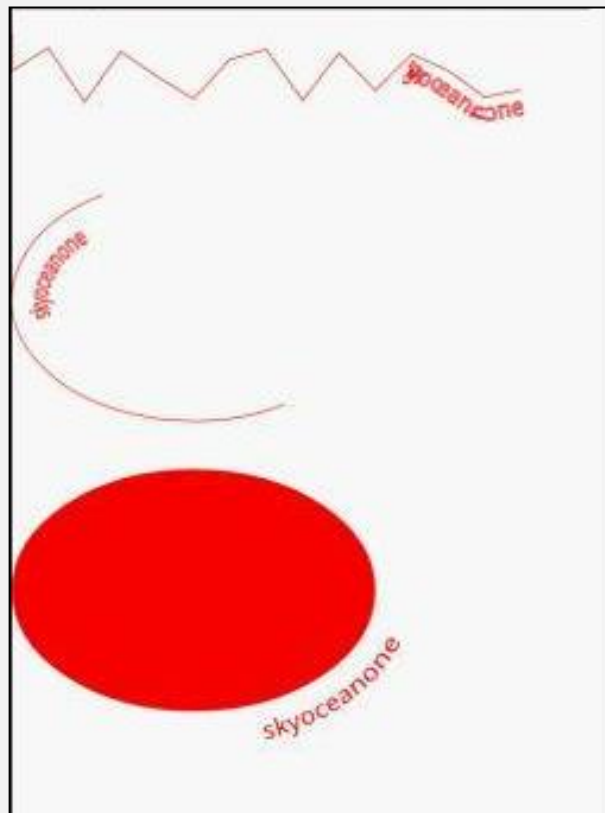
- Path类可以预先在View中连成路径，然后调用Canvas的drawPath在画布上画出。
- 通过setPathEffect来给Path增加绘制效果。
- PathEffect的常见子类：
 - ComposePathEffect
 - CornerPathEffect
 - DashPathEffect
 - PathDashPathEffect
 - SumPathEffect





Android中的绘图

- 通过Canvas的drawTextOnPath方法可以在画布上沿着某条Path绘制文本。

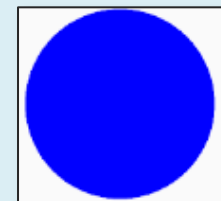




Android中的绘图

- 画纯色填充的圆

```
Bitmap bitmap = Bitmap.createBitmap(width, height,  
Bitmap.Config.ARGB_8888);  
Canvas canvas = new Canvas(bitmap);  
Paint paint = new Paint();  
Path path = new Path();  
paint.setStrokeWidth(5);  
paint.setColor(Color.BLUE);  
canvas.drawCircle(400, 400, 150, paint);  
canvas.drawPath(path, paint);  
ivImg.setImageBitmap(bitmap);
```

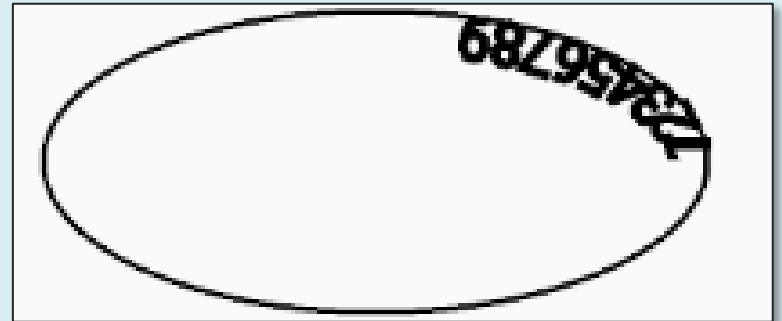




Android中的绘图

- 画内部文字的椭圆

```
Bitmap bitmap = Bitmap.createBitmap(width, height,  
Bitmap.Config.ARGB_8888);  
Canvas canvas = new Canvas(bitmap);  
Paint paint = new Paint();  
Path path = new Path();  
paint.setStrokeWidth(5);
```





Android中的绘图

- 画内部文字的椭圆（续）

```
paint.setColor(Color.BLACK);  
paint.setStyle(Paint.Style.STROKE);  
path.addOval(50, 350, 600, 600, Path.Direction.CCW);  
canvas.drawPath(path, paint);  
paint.setTextSize(50);  
// 绘制路径上的文字  
canvas.drawTextOnPath("123456789", path, 0, 0, paint);  
ivImg.setImageBitmap(bitmap);
```



课后练习

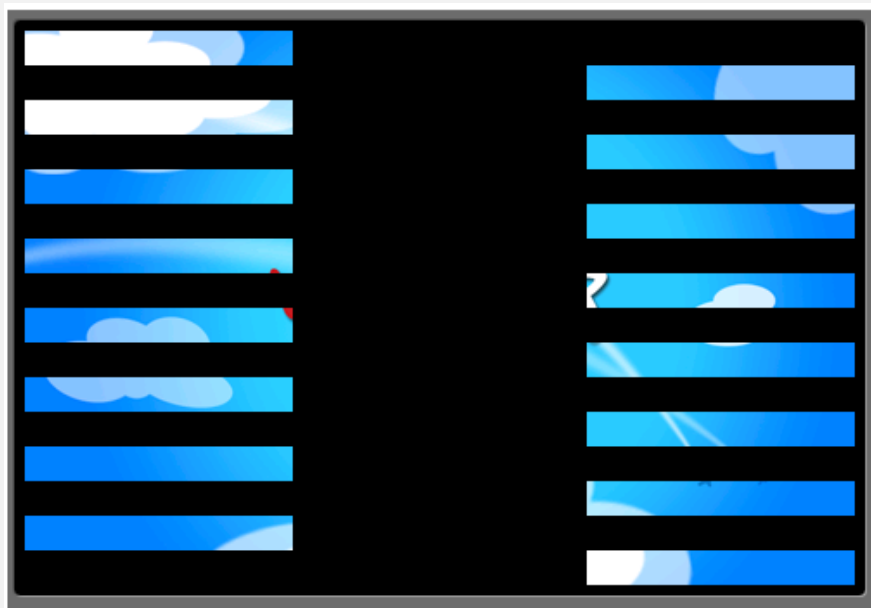


- http://www.oschina.net/question/231733_44154
- <http://blog.csdn.net/eastmount/article/details/40689397>



绘图在游戏中的作用（补）

- 游戏中的动画就是不断的调用View中的onDraw方法，在画布上依次绘制不同的图像，从而形成了动画。





目录

- 1 Android中简单的图片使用
- 2 Android中的绘图
- 3 Android中的图形特效**



Android中的图形特效

- Android中还提供了一些更高级的图形特效支持，这些特效可以让我们画出更加绚丽的UI。
- 常见特效：
 - 使用Matrix矩阵控制图像变换。
 - 使用drawbitmapmesh扭曲图像。
 - 使用shader填充图形。



Android中的图形特效

- Matrix的对图像的处理可分为四类基本变换。

英文	中文
Translate	平移变换
Rotate	旋转变换
Scale	缩放变换
Skew	错切变换



使用Matrix矩阵控制图像变换

- Matrix是一个矩阵工具类，它配合其他图形类来控制图形和组件的变换。
- 使用Matrix控制图形或组件变换的步骤如下：
 - 获取Matrix对象：可新创建，也可从其他对象内获取；
 - 调用Matrix的方法进行平移，旋转，缩放，倾斜等；
 - 将程序对Matrix做的变换应用到图像或者组件。



使用Matrix矩阵控制图像变换

将图片旋转60度:



图片倾斜: `mMatrix.postSkew(0.3f, 0.7f)`;效果:



图片缩放, x轴缩小0.5倍, y轴扩大2.5倍: `mMatrix.setScale(0.5f, 2.5f)`;效果:





使用Matrix矩阵控制图像变换

- Matrix类常用的方法：

- `set(Matrix src) : void - Matrix`
- `setConcat(Matrix a, Matrix b) : boolean - Matrix`
- `setPolyToPoly(float[] src, int srcIndex, float[] dst, int dstIndex, int pointCount) : boolean - Matrix`
- `setRectToRect(RectF src, RectF dst, ScaleToFit stf) : boolean - Matrix`
- `setRotate(float degrees) : void - Matrix`
- `setRotate(float degrees, float px, float py) : void - Matrix`
- `setScale(float sx, float sy) : void - Matrix`
- `setScale(float sx, float sy, float px, float py) : void - Matrix`
- `setSinCos(float sinValue, float cosValue) : void - Matrix`
- `setSinCos(float sinValue, float cosValue, float px, float py) : void - Matrix`
- `setSkew(float kx, float ky) : void - Matrix`
- `setSkew(float kx, float ky, float px, float py) : void - Matrix`
- `setTranslate(float dx, float dy) : void - Matrix`
- `setValues(float[] values) : void - Matrix`



使用Matrix矩阵控制图像变换

- 将Matrix应用到图形或组件上：
 - `Bitmap.createBitmap(Bitmap, int , int, int,int, Matrix,boolean)`
 - `Canvas`类的`drawBitmap (Bitmap,Matrix,Paint) ;`
 -



使用Matrix矩阵控制图像变换

- 平移变换

```
//平移变化  
Matrix matrix1 = new Matrix();  
//设置应用Matrix变化形式  
ivChange.setScaleType(ImageView.ScaleType.MATRIX);  
//或在XML布局文件中对控件设置android:scaleType="matrix"  
matrix1.postTranslate(700, ivOrigin.getHeight()+20);  
ivChange.setImageMatrix(matrix1);
```



使用Matrix矩阵控制图像变换

- 旋转变换

```
LinearLayout.LayoutParams params = new  
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH  
_PARENT, ViewGroup.LayoutParams.MATCH_PARENT);  
ivChange.setLayoutParams(params);  
ivChange.setImageResource(R.mipmap.fj4);  
ivChange.setScaleType(ImageView.ScaleType.MATRIX);  
Matrix matrix2 = new Matrix(); // 创建matrix对象  
matrix2.setRotate(-45); // 旋转图片动作  
System.out.println(ivOrigin.getHeight());  
matrix2.postTranslate(200, 200);  
ivChange.setImageMatrix(matrix2);
```



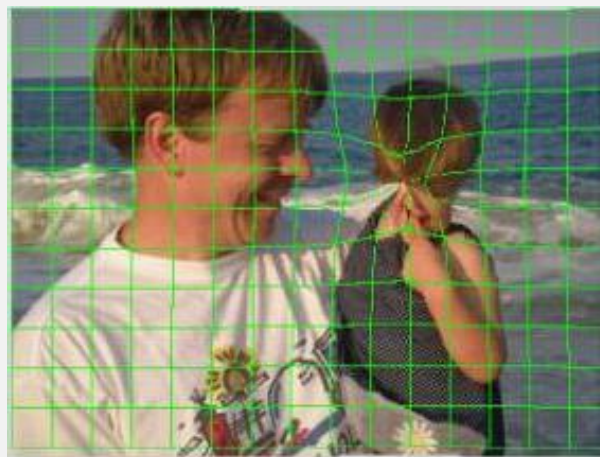
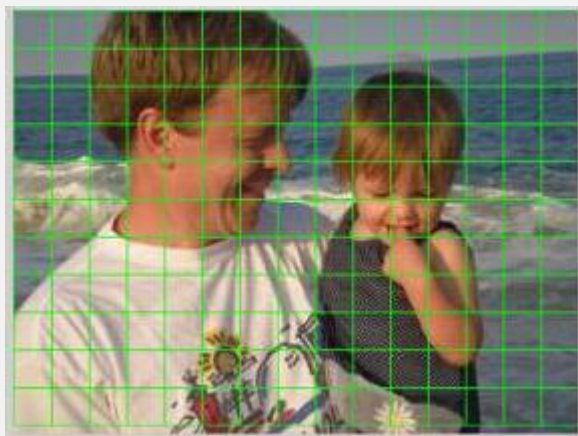
Android中的图形特效

- 常见特效：
 - 使用Matrix矩阵控制图像变换。
 - 使用drawBitmapMesh扭曲图像。
 - 使用shader填充图形。



使用drawbitmapmesh扭曲图像

- Canvas提供了drawBitmapMesh (Bitmap,..) 方法实现图片的扭曲，例如：水波荡漾，风吹红旗的扭曲效果。





使用drawbitmapmesh扭曲图像

- drawBitmapMesh(Bitmap bitmap, int meshWidth, int meshHeight, float[] verts, int vertOffset, int[] colors, int colorOffset, Paint paint)
 - bitmap 需要转换的位图
 - meshWidth 横向的格数，需大于 0
 - meshHeight 纵向的格数，需大于 0
 - verts 网格顶点坐标数组，记录扭曲后图片各顶点的坐标，数组大小为 $(meshWidth+1) * (meshHeight+1) * 2 + vertOffset$
 - vertOffset 从第几个顶点开始对位图进行扭曲，通常传 0
 - colors 设置网格顶点的颜色，该颜色会和位图对应像素的颜色叠加，数组大小为 $(meshWidth+1) * (meshHeight+1) + colorOffset$ ，可以传 null
 - colorOffset 从第几个顶点开始转换颜色，通常传 0
 - paint 「画笔」，可以传 null

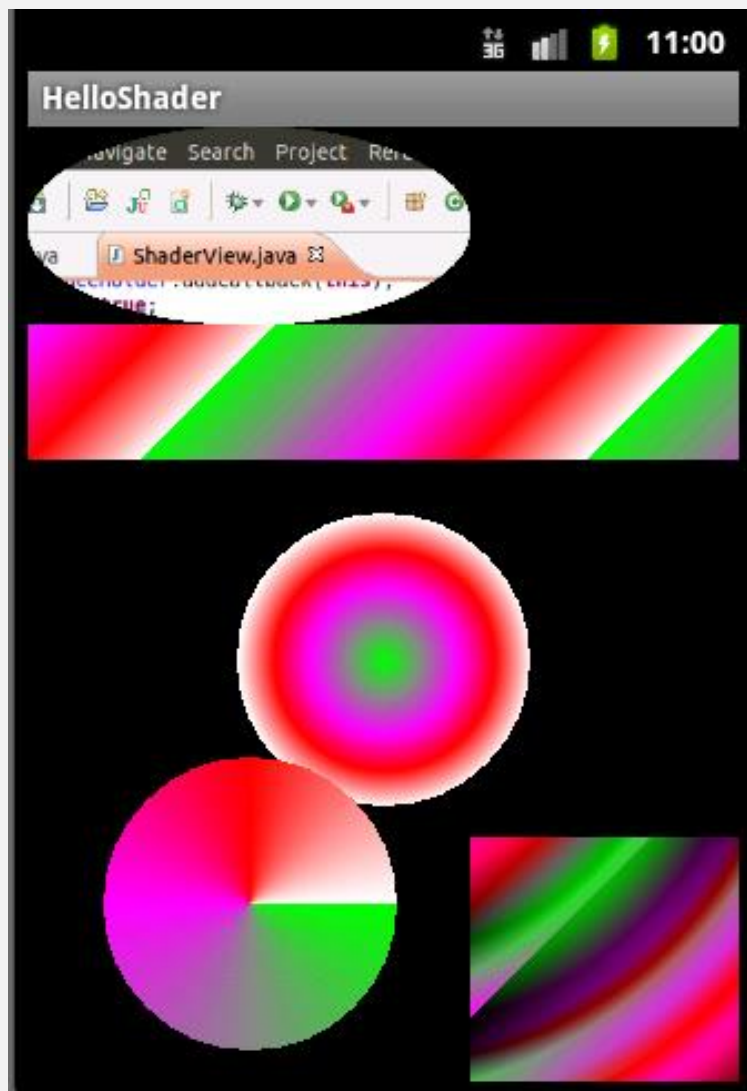


使用shader填充图形

- 在通过Paint类画图的过程中可以通过setShader给画笔设置相应的渲染效果，当然在绘制Path时，也可以通过Shader对象来填充绘制的图形。
- Shader是一个抽象类，常见的实现类：
 - BitmapShader：图像渲染
 - LinearGradient：线性渐变
 - RadialGradient：环形渐变
 - SweepGradient：扫描渐变---围绕一个中心点扫描渐变就像电影里那种雷达扫描
 - ComposeShader：组合渲染



使用shader填充图形





课程回顾

- 1 Android中简单的图片使用
- 2 Android中的绘图
- 3 Android中的图形特效



Thank you!

