图形与图像处理

1.1.1 使用Drawable对象

1.1.2 Bitmap和BitmapFactory

Bitmap代表一个位图，BitmapDrawable里封装的图片就是一个Bitmap对象，开发者为了把一个Bitmap对象包装成BitmapDrawable对象，可以调用BitmapDrawable的构造器。

```
//把一个Bitmap对象包装成BitmapDrawable对象
BitampDrawable drawable = BitmapDrawable(bitmap)
```

如果需要获取BitmapDrawable所包装的Bitmap对象，则可调用BitmapDrawable的getBitmap()方法，如下面代码所示。

```
//获取BitmapDrawable所包装的Bitmap对象
Bitmap bitmap = drawable.getBitmap();
```

另外，Bitmap还提供了一些静态方法来创建新的Bitmap对象，例如如下常用方法。

- createBitmap(Bitmap source, int x ,int y,int width,int height):从源位图source指定的坐标点（给定x、y）开始，从中挖取宽width、高height的一块出来创建新的Bitmap对象。
- createBitmap(int width,int height，Bitmap.Config config):创建一个宽width、高height的新位图。
- createBitmap(Bitmap source, int x ,int y,int width,int height，Matrix m, boolean filter)：从源位图source的指定坐标点（给定x、y）开始，从中“挖取”宽width、高height的一块出来，创建新的Bitmap对象，并按照Matrix指定的规则进行变换；

BitmapFactory是一个工具类，它提供了大量的方法，这些方法可用于从不同的数据源来解析、创建Bitmap对象。BitmapFactory包含了如下方法。

- decodeByteArray(byte[] data,int offset,int length):从指定字节数组的offset位置开始，将长度为length的字节数据解析成Bitmap对象。
- decodeFile（String pathName）：从pathName指定的文件中解析、创建Bitmap对象。
- decodeFileDescriptor(FileDescriptor fd):用于从FileDescriptor对应的文件中解析、创建Bitmap对象。
- decodeResource（Resource res，int id）：用于根据给定的资源ID从指定资源中解析、创建Bitmap对象。
- decodeStream（InputStream is）：用于从指定输入流中解析、创建Bitmap对象。

Android为Bitmap提供了两个方法来判断它是否已回收，以强制Bitmap回收自己。

- boolean isRecyled():返回该Bitmap对象是否已被回收。
- void recyle()：强制一个Bitmap对象立即回收自己。

1.1.3 Android新增的ImageDecoder

不仅可以解码PNG\JPEG等静态图片，而且也能直接解码GIF\WEBP等动画图片。

此外，Android9新增了支持HEIF格式，这种格式具有超高的压缩比，相较JPEG格式，其文件大小可以压缩到只有其一半，且可保证近似的图像质量。

使用ImageDecoder解码GIF、WEBP等动画图片时，程序将会返回一个AnimatedImageDrawable对象，为了开始执行动画，调用AnimatedImageDrawable对象的start()方法即可。

使用ImageDecider解码图片很简单，只要如下两步即可。

- 调用ImageDecoder的重载的createSource方法来创建Source对象。根据不同的图片来源，createSource方法有不同的重载形式。
- 调用ImageDecoder的decodeDrawable（Source） or decodeBitmap（Source） 方法来读取代表图片的Drawable或Bitmap对象。

在执行上面第2步时，程序可以额外传入一个OnHeaderDecoderListener参数，该参数代表一个监听器，该监听器要实现一个onHeaderDecoder（ImageDecoder, ImageInfo,Source）方法，通过该方法可以对ImageDecoder进行额外的设置，也可以通过ImageInfo获取被解码图片的信息。

1.2 绘图

1.2.1 Android绘图基础：Canvas、Paint等

绘图，继承View组件，并重写它的onDraw方法即可；

canvas：代表“依附”于指定View的画布；其中Path：代表任意多条直线连接而成的任意图形。

1.2.3 绘制游戏动画

1.3 图形特效处理

1.3.1使用Matrix控制变换

动画

动画分类

补间动画，帧动画，以及最新的属性动画。

帧动画就是类似电影，由多张图片切换而成。

补间动画可以望文生义吧，就是在两点之间插入渐变值来平滑过渡。

属性动画和补间动画类似，不过是真的属性在变动，包括可视属性和其他属性。

补间动画仅仅是可视属性在显示层面的动画，属性的实质并未改动。

一般情况下推荐使用最新的属性动画。

1.4 逐帧（Frame）动画

1.5 补间（Tween）动画

Android使用Animation代表抽象的动物类，包括了如下几个子类。

- AlphaAnimation：透明度改变的动画；
- ScaleAnimation：大小缩放的动画；
- TranslateAnimation：位移变化的动画；
- RoatateAnimation：旋转动画；

  一旦为补间动画指定了三个必要信息，Android就会根据动画的开始帧、结束帧、动画持续时间计算出需要在中间“补入”多少帧，计算所有补入帧的图形。当用户浏览补间动画时，眼中依然是“逐帧动画”。

  为了控制在动画期间需要动态“补入”多少帧，具体在动画运行的哪些时刻补入帧，需要借助于Interpolator。

Interpolator提供了如下几个实现类;

- LinearIterpolator: 动画以匀速的速度改变；
- AccelerateInterpolator：在动画开始的地方改变速度较慢，然后开始加速；
- AccelerateDecelerateInterpolator：在动画开始、结束的时候较慢，中间的时候较快；
- CycleInterpolator：动画循环播放特定的次数，变化速度按正弦曲线改变；
- DecelerateInterpolator：动画开始的地方变化快，然后开始减速；