

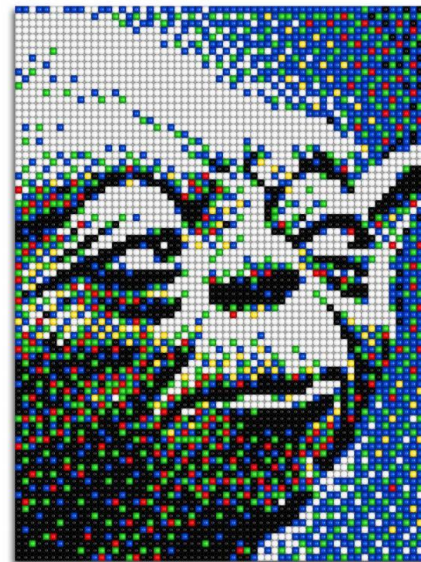
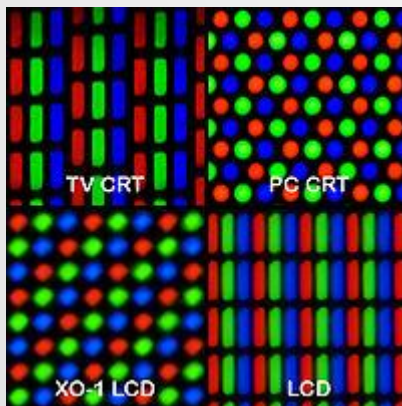
A faint, light gray world map is visible in the background, centered on the Atlantic Ocean. The map shows the outlines of continents and major landmasses.

ANDROID移动互联网应用开发

傅晓，河海大学计算机与信息学院
2019年3月

安卓常用绘图类

- Bitmap: 相当于我们绘制出来的图像, 获取图像文件信息, 对图像进行剪切、旋转、缩放, 压缩等操作, 并可以以指定格式保存图像文件。
- Paint: 相当于我们绘图所用的画笔, 绘制几何, 文本, 位图的风格与颜色信息。
- Canvas: 相当于我们绘图所用的画布, 调用方法进行图像绘制。

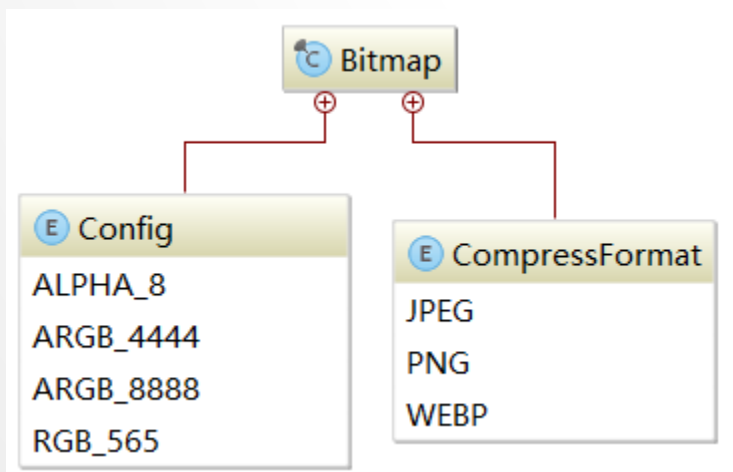


BITMAP

- Bitmap在Android中指的是一张图片，可以是png，也可以是jpg等其他图片格式。
- Bitmap是Android系统中的图像处理中最重要类之一。Bitmap可以获取图像文件信息，对图像进行剪切、旋转、缩放，压缩等操作，并可以以指定格式保存图像文件。
- Bitmap是一个final类，因此不能被继承。Bitmap只有一个构造方法，且该构造方法是没有任何访问权限修饰符修饰，也就是说该构造方法是friendly。

BITMAP

- Bitmap中有两个内部枚举类：
- Config是用来设置颜色配置信息的。
- CompressFormat是用来设置压缩方式的。



BITMAP










- Bitmap.Config.ALPHA_8: 颜色信息只由透明度组成, 占8位。
- Bitmap.Config.ARGB_4444: 颜色信息由透明度与R (Red), G (Green), B (Blue) 四部分组成, 每个部分都占4位, 总共占16位。
- Bitmap.Config.ARGB_8888: 颜色信息由透明度与R (Red), G (Green), B (Blue) 四部分组成, 每个部分都占8位, 总共占32位。
是Bitmap默认的颜色配置信息, 也是最占空间的一种配置。
- Bitmap.Config.RGB_565: 颜色信息由R (Red), G (Green), B (Blue) 三部分组成, R占5位, G占6位, B占5位, 总共占16位。












BITMAP

- `Bitmap.CompressFormat.JPEG`: 表示以JPEG压缩算法进行图像压缩, 压缩后的格式可以是".jpg"或者".jpeg", 是一种有损压缩。
- `Bitmap.CompressFormat.PNG`: 表示以PNG压缩算法进行图像压缩, 压缩后的格式可以是".png", 是一种无损压缩。
- `Bitmap.CompressFormat.WEBP`: 表示以WebP压缩算法进行图像压缩, 压缩后的格式可以是".webp", 是一种有损压缩, 质量相同的情况下, WebP格式图像的体积要比JPEG格式图像小40%。美中不足的是, WebP格式图像的编码时间“比JPEG格式图像长8倍”。

BITMAP

- Bitmap的类构造函数是私有的，因此不能直接通过构造方法实例化。
- 一般利用Bitmap的静态方法createBitmap()和BitmapFactory的decode系列静态方法创建Bitmap对象。

Bitmap		
	createBitmap(Bitmap)	Bitmap
	createBitmap(Bitmap, int, int, int, int)	Bitmap
	createBitmap(Bitmap, int, int, int, int, Matrix, boolean)	
	createBitmap(int, int, Config)	Bitmap
	createBitmap(DisplayMetrics, int, int, Config)	Bitmap
	createBitmap(int[], int, int, int, int, Config)	Bitmap
	createBitmap(DisplayMetrics, int[], int, int, int, int, Config)	
	createBitmap(int[], int, int, Config)	Bitmap
	createBitmap(DisplayMetrics, int[], int, int, Config)	map

BitmapFactory		
	decodeFile(String, Options)	Bitmap
	decodeFile(String)	Bitmap
	decodeResourceStream(Resources, TypedValue, InputStream, Rect)	
	decodeResource(Resources, int, Options)	Bitmap
	decodeResource(Resources, int)	Bitmap
	decodeByteArray(byte[], int, int, Options)	Bitmap
	decodeByteArray(byte[], int, int)	Bitmap
	decodeStream(InputStream, Rect, Options)	Bitmap
	decodeStream(InputStream)	Bitmap
	decodeFileDescriptor(FileDescriptor, Rect, Options)	Bitmap
	decodeFileDescriptor(FileDescriptor)	Bitmap

BITMAP

- BitmapFactory类提供了4类方法用来加载Bitmap:
- decodeFile(): 从文件系统加载。
- String sd_patch = “/sdcard/test.png” ;
- Bitmap bm = BitmapFactory.decodeFile(sd_path);
- decodeResource(): 以R.drawable.xxx的形式从本地资源中加载。
- Bitmap bm =
BitmapFactory.decodeResource(this.getContext().getResources(
, R.drawable.pop);

BITMAP

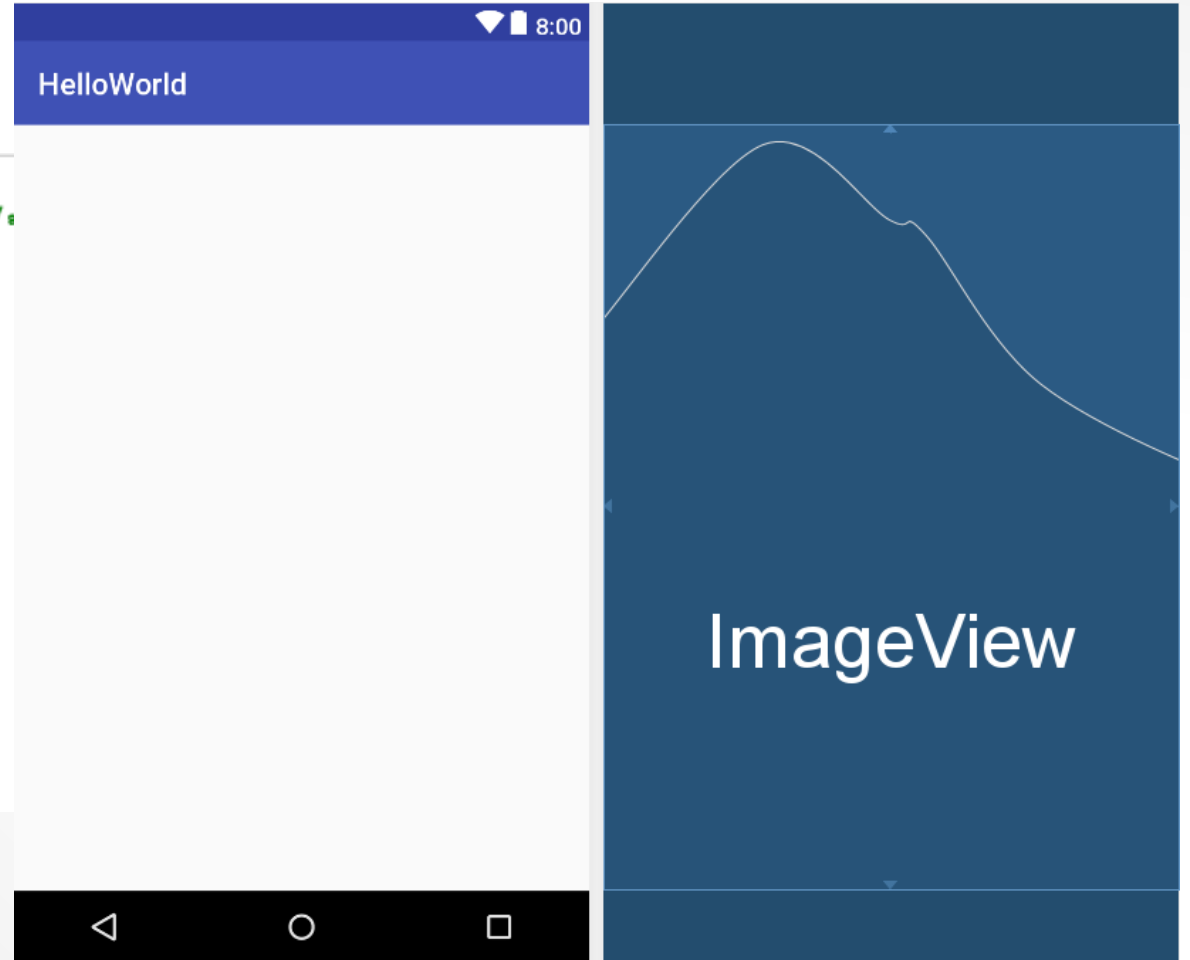
- `decodeStream()`: 从输入流加载。
- `FileInputStream = new FileInputStream("/sdcard/test.png");`
- `Bitmap bm = BitmapFactory.decodeStream(fis);`
- `decodeByteArray()`: 从字节数组中加载。
- `Bitmap bm =
 BitmapFactory.decodeByteArray(myByte,0,myByte.length);`

BITMAP

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/s
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.bishop.helloworld.MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

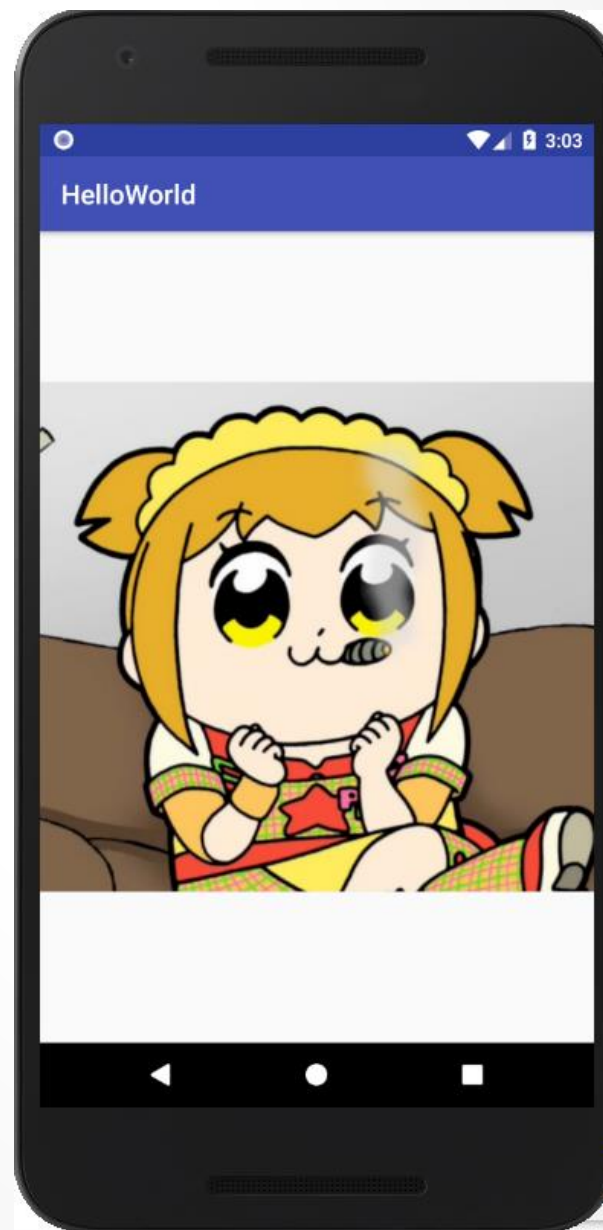


BITMAP

```
package com.example.bishop.helloworld;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {
    ImageView img = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img=(ImageView)findViewById(R.id.imageView);
        Bitmap bm = BitmapFactory.decodeResource(this.getResources(), R.drawable.pop);
        img.setImageBitmap(bm);
    }
}
```



PAINT

- Paint有3个构造方法，可以通过这3个构造方法创建Paint对象：
- Paint()：用默认设置创建一个Paint对象。
- Paint(int flags)：用特殊标记创建一个Paint对象：
- Paint.FILTER_BITMAP_FLAG：使位图过滤的位掩码标志。
- Paint.ANTI_ALIAS_FLAG：使位图抗锯齿的标志。
- Paint.DITHER_FLAG：使位图进行有利的抖动的位掩码标志。
- Paint(Paint paint)：用指定Paint对象的参数初始化一个新的Paint对象。

PAINT

- Paint常用方法：
- setARGB(int a, int r, int g, int b): 设置画笔颜色。
- setAntiAlias(boolean aa): 设置是否抗锯齿。
- setColor(int color): 设置画笔颜色。
- setAlpha(int a): 设置画笔透明度。
- setTextSize(float textSize): 设置字体大小。
- setUnderlineText(boolean underlineText): 设置文本带下划线效果。
- setStrikeThruText(boolean strikeThruText): 设置文本带删除线效果。

PAINT

- `setTextSkewX(float skewX)`: 设置文本倾斜度。
- `setTextScaleX(float scaleX)`: 设置文本缩放大小。
- `setTextAlign(Paint.Align align)`: 设置文本对齐方式。
- `setTypeface(Typeface typeface)`: 设置字体。
- `setStyle(Paint.Style style)`: 设置画笔样式，画笔样式有3种：
 `Paint.Style.FILL`: 默认值，用这种风格绘制的几何图与文本将被填充，它画出来的是实心图
- `Paint.Style.STROKE`: 用这种风格绘制的几何图与文本将被画出外边框，它画出来的是空心图
- `Paint.Style.FILL AND STROKE`: 用这种风格绘制的几何图与文本将被填充并被画出外边框，从表面看它画出来的也是实心图，不过比一般画出来的实心图多了一层外边框。

PAINT

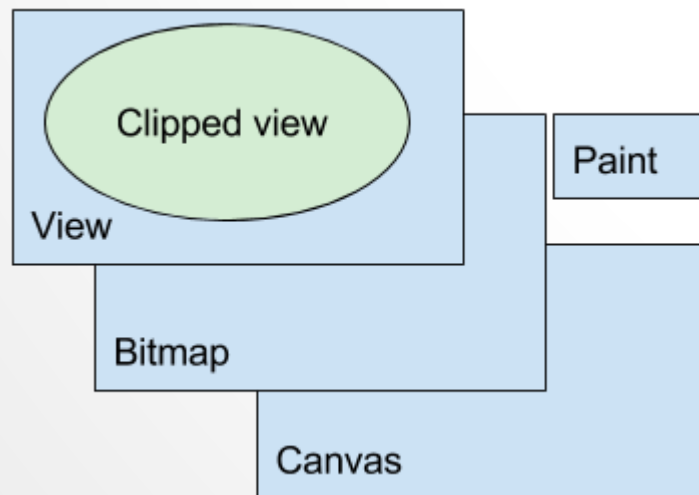
- `setStrokeWidth(float width)`: 设置画笔外边框的宽度，可以想象成画笔“画出线条的宽度”。
- `setXfermode(Xfermode xfermode)`: 设置图像重叠时的处理方式。
- `setShader(Shader shader)`: 设置着色器。
- `setPathEffect(PathEffect effect)`: 设置或者清除路径效果。

CANVAS

- Canvas拥有“绘制”调用，可以调用方法进行图像绘制。
- 画图时需要4个基本元素：
- 拥有像素的Bitmap。
- 可以进行绘制调用的Canvas。
- 图元（比如：Rect, Path, text, Bitmap）。
- 描述风格与颜色的Paint。

CANVAS

- Canvas有两个构造方法，可以通过这两个构造方法创建Canvas对象：
- Canvas()：创建一个空的Canvas对象。
- Canvas(Bitmap bitmap)：用指定的位图构造一个Canvas对象。



CANVAS

- Canvas常用方法：
- drawARGB(): 用指定ARGB颜色填充画布上面的位图
- drawRGB(): 用指定RGB颜色填充画布上面的位图
- drawColor(): 用指定颜色填充画布上面的位图
- drawArc(): 画圆弧
- drawBitmap(): 画位图
- drawCircle(): 画圆
- drawLine(): 画直线

CANVAS

- `drawLines()`: 画折线
- `drawOval()`: 画椭圆
- `drawRect()`: 画矩形
- `drawRoundRect()`: 画圆角矩形
- `drawPoint()`: 画点
- `drawPoints()`: 画一组点
- `drawPath()`: 画路径
- `drawText()`: 画文本

CANVAS

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.bishop.helloworld.MainActivity">
<com.example.bishop.helloworld.MyView
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</android.support.constraint.ConstraintLayout>
```

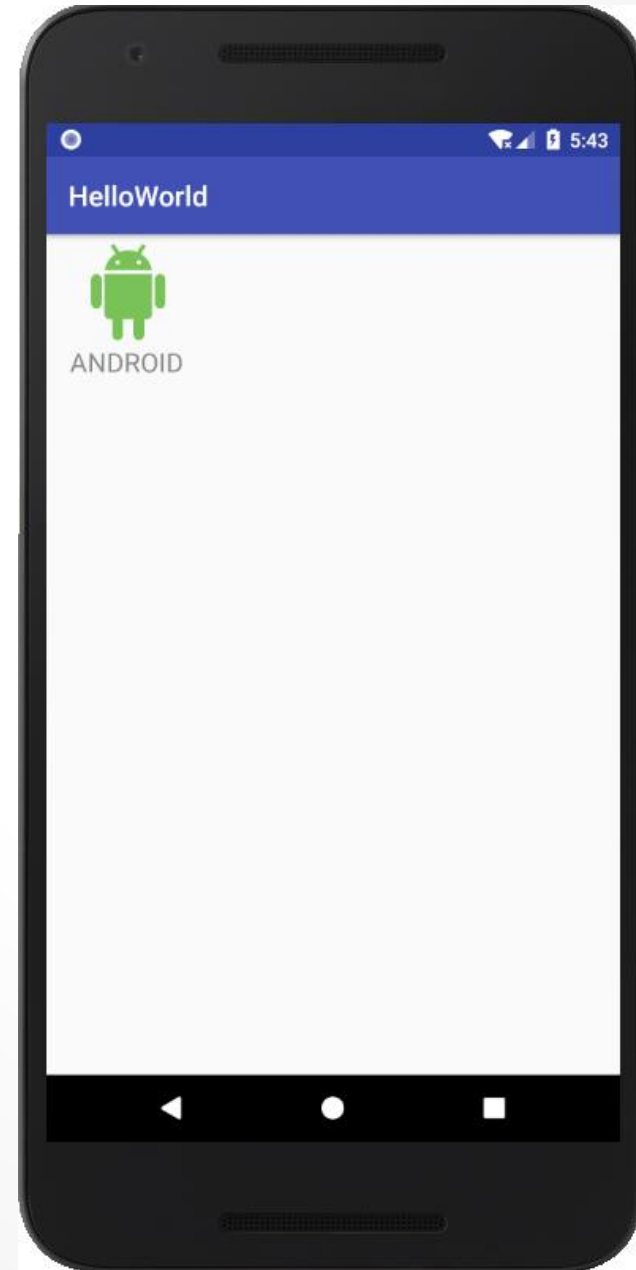
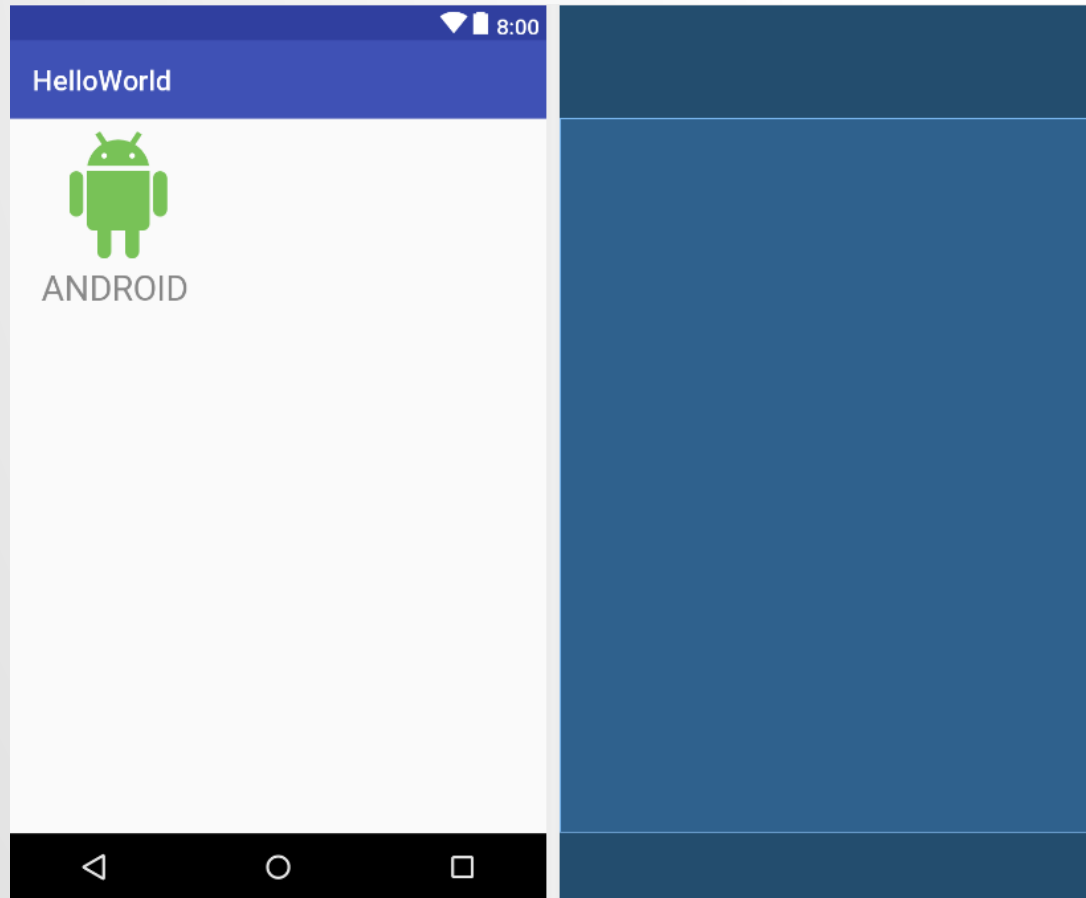
```
package com.example.bishop.helloworld;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.util.AttributeSet;
import android.view.View;
```

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

    @Override
    protected void onDraw(Canvas canvas){
        Paint paint=new Paint();
        paint.setAntiAlias(true);
        paint.setColor(0xFF78C257);
        RectF rectf_head=new RectF( left: 110, top: 30, right: 200, bottom: 120);
        canvas.drawArc(rectf_head, startAngle: -10, sweepAngle: -160, useCenter: false, paint);
        paint.setColor(Color.WHITE);
        canvas.drawCircle( cx: 135, cy: 53, radius: 4, paint);
        canvas.drawCircle( cx: 175, cy: 53, radius: 4, paint);
        paint.setColor(0xFF78C257);
        paint.setStrokeWidth(8);
        canvas.drawLine( startX: 125, startY: 20, stopX: 135, stopY: 35, paint);
        canvas.drawLine( startX: 185, startY: 20, stopX: 175, stopY: 35, paint);
        canvas.drawRect( left: 110, top: 75, right: 200, bottom: 150, paint);
        RectF rectf_body=new RectF( left: 110, top: 140, right: 200, bottom: 160);
        canvas.drawRoundRect(rectf_body, rx: 10, ry: 10, paint);
        RectF rectf_arm=new RectF( left: 85, top: 75, right: 105, bottom: 140);
        canvas.drawRoundRect(rectf_arm, rx: 10, ry: 10, paint);
        rectf_arm.offset( dx: 120, dy: 0);
        canvas.drawRoundRect(rectf_arm, rx: 10, ry: 10, paint);
        RectF rectf_leg=new RectF( left: 125, top: 150, right: 145, bottom: 200);
        canvas.drawRoundRect(rectf_leg, rx: 10, ry: 10, paint);
        rectf_leg.offset( dx: 40, dy: 0);
        canvas.drawRoundRect(rectf_leg, rx: 10, ry: 10, paint);
        paint.setTextSize(50);
        paint.setColor(Color.GRAY);
        canvas.drawText( text: "ANDROID", x: 45, y: 260, paint);
        super.onDraw(canvas);
    }
}
```

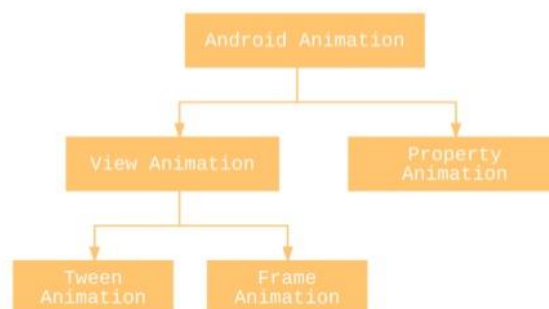
CANVAS



ANIMATION

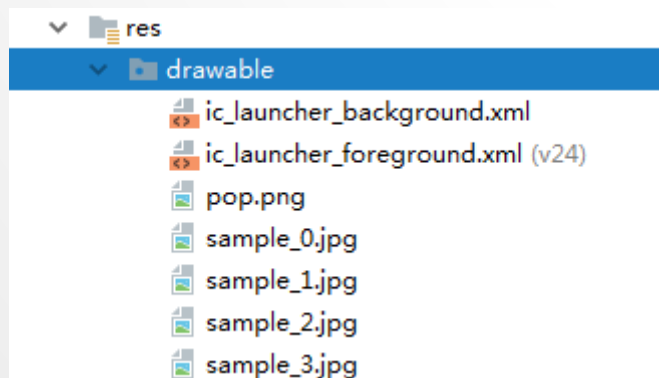
- Android Animation分为两类：
- 传统动画：包括帧动画（Frame Animation）和补间动画（Tween Animation），又称为Drawable Animation和View Animation。
- 属性动画（Property Animation）。

Animation in Android

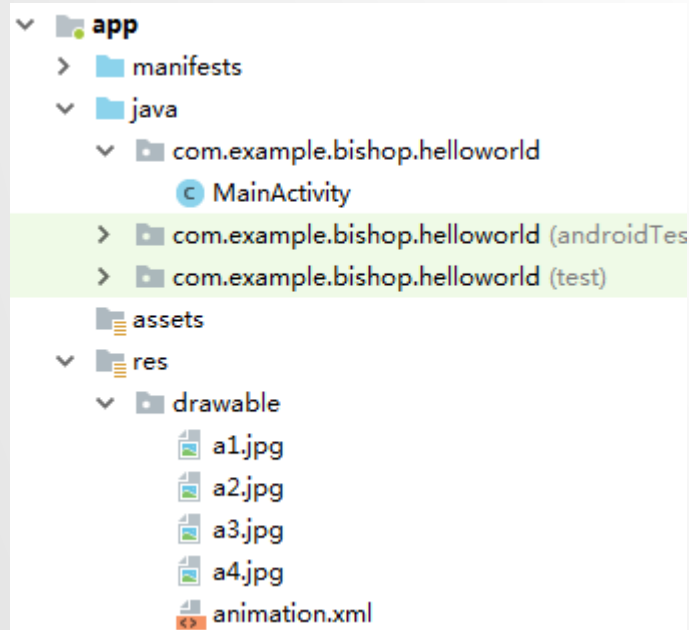


ANIMATION

- Frame Animation (Drawable Animation) 是最容易实现的一种动画，这种动画更多的依赖于完善的UI资源。
- Frame Animation的原理是将一张张单独的图片连贯的进行播放，从而在视觉上产生一种动画的效果。
- 这种动画的实质其实是Drawable，所以这种动画的XML定义方式文件一般放在res/drawable/目录下。



ANIMATION



```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/a1"
        android:duration="200"/>
    <item android:drawable="@drawable/a2"
        android:duration="200"/>
    <item android:drawable="@drawable/a3"
        android:duration="200"/>
    <item android:drawable="@drawable/a4"
        android:duration="200"/>
</animation-list>
```



a1.jpg



a2.jpg



a3.jpg



a4.jpg



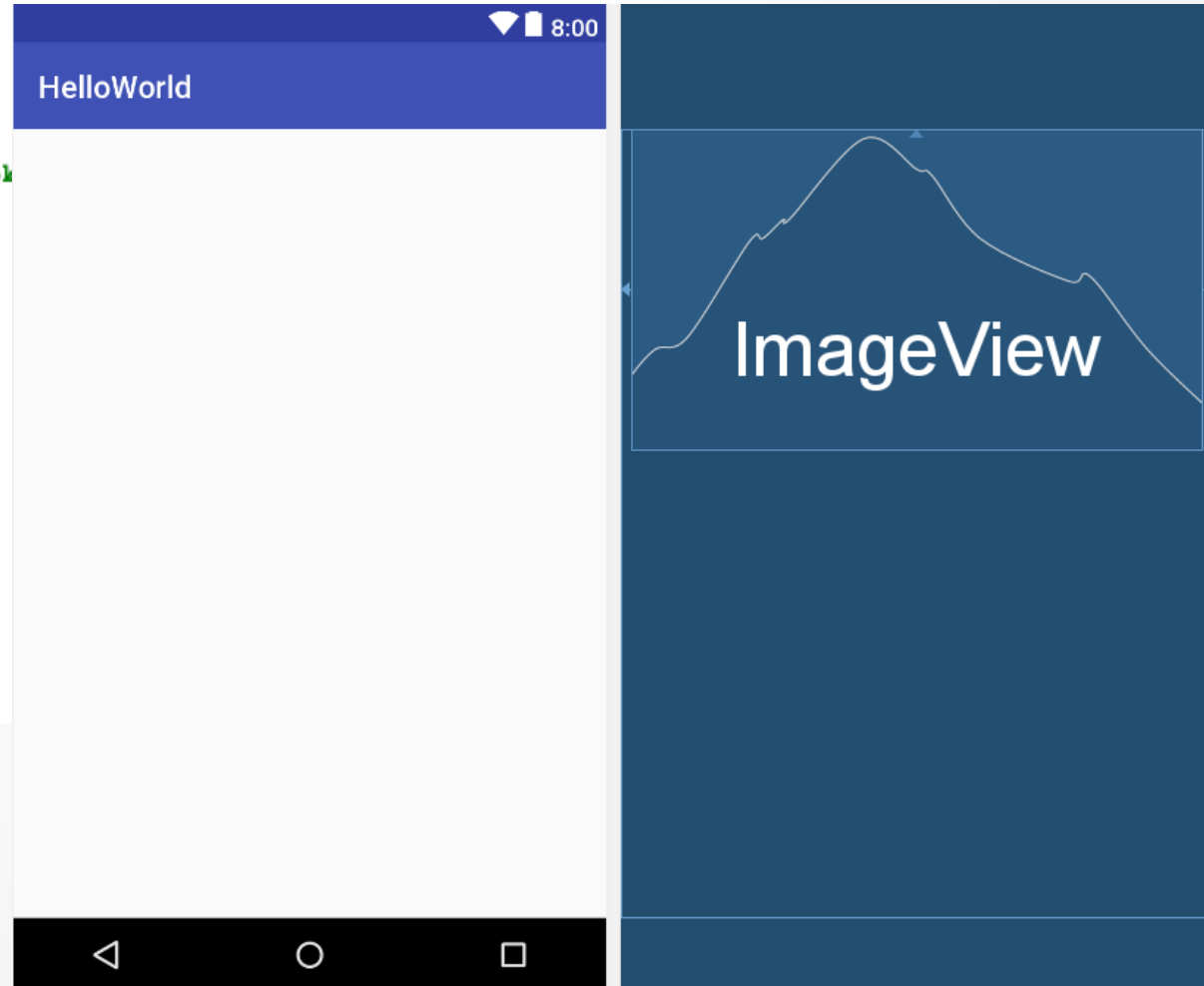
animation.xml

ANIMATION

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.bishop.helloworld.MainActivity">

    <ImageView
        android:id="@+id/animation"
        android:layout_width="370dp"
        android:layout_height="208dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



ANIMATION

```
package com.example.bishop.helloworld;

import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {
    ImageView img = null;
    AnimationDrawable animation;

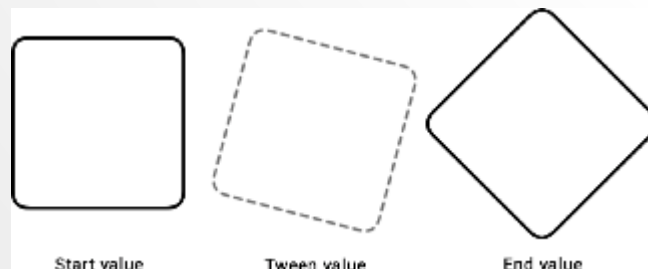
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onResume() {
        super.onResume();
        img = (ImageView) findViewById(R.id.animation);
        img.setBackgroundResource(R.drawable.animation);
        animation = (AnimationDrawable) MainActivity.this.img.getBackground();
        animation.start();
    }
}
```



ANIMATION

- Tween Animation (View Animation) 可以在一个视图容器内执行一系列简单变换。譬如，有一个TextView对象，可以移动、旋转、缩放、透明度设置其文本，当然，如果它有一个背景图像，背景图像会随着文本变化。
- Tween Animation 具体有4种形式：alpha（淡入淡出），translate（位移），scale（缩放大小），rotate（旋转）。
- 补间动画通过XML或Android代码定义，建议使用XML文件定义，因为它更具可读性、可重用性。



ANIMATION

- 相关类名：
- AlphaAnimation：渐变透明度动画效果，对应XML中的<alpha>，放置在res/anim/目录。
- RotateAnimation：画面转移旋转动画效果，对应XML中的<rotate>，放置在res/anim/目录。
- ScaleAnimation：渐变尺寸伸缩动画效果，对应XML中的<scale>，放置在res/anim/目录。
- TranslateAnimation：画面转换位置移动动画效果，对应XML中的<translate> 放置在res/anim/目录。
- AnimationSet：一个持有其它动画元素alpha、scale、translate、rotate 或者其它set元素的容器，对应XML中的<set>，放置在res/anim/目录。

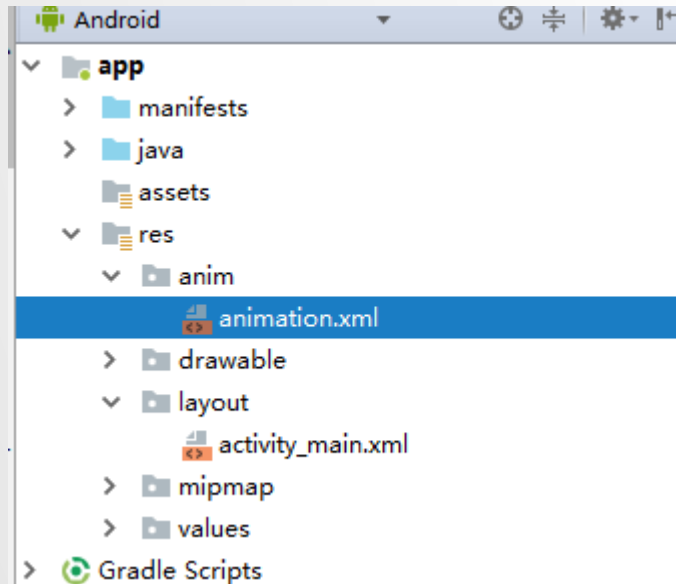
ANIMATION

- Animation属性:
- android:detachWallpaper: 是否在壁纸上运行, 对应setDetachWallpaper(boolean)。
- android:duration: 动画持续时间, 毫秒为单位, 对应setDuration(long)。
- android:fillAfter: 控件动画结束时是否保持动画最后的状态, 对应setFillAfter(boolean)。
- android:fillBefore: 控件动画结束时是否还原到开始动画前的状态, 对应setFillBefore(boolean)。
- android:fillEnabled: 与android:fillBefore效果相同, 对应setFillEnabled(boolean)。
- android:interpolator: 设定插值器 (指定的动画效果, 譬如回弹等), 对应setInterpolator(Interpolator)。
- android:repeatCount: 重复次数, 对应setInterpolator(Interpolator)。
- android:repeatMode: 重复类型有两个值, reverse表示倒序回放, restart表示从头播放, 对应setInterpolator(Interpolator)。
- android:startOffset: 调用start函数之后等待开始运行的时间, 单位为毫秒, 对应setStartOffset(long)。
- android:zAdjustment: 表示被设置动画的内容运行时在Z轴上的位置 (top/bottom/normal), 默认为normal, 对应setZAdjustment(int)。

ANIMATION

- Interpolator 主要作用是可以控制动画的变化速率，就是动画进行的快慢节奏：
- @android:anim/accelerate_decelerate_interpolator：动画始末速率较慢，中间加速。
- @android:anim/accelerate_interpolator：动画开始速率较慢，之后慢慢加速。
- AnticipateInterpolator @android:anim/anticipate_interpolator 开始的时候从后向前甩。
- @android:anim/anticipate_overshoot_interpolator 类似上面AnticipateInterpolator。
- @android:anim/bounce_interpolator：动画结束时弹起。
- @android:anim/cycle_interpolator：循环播放速率改变为正弦曲线。
- @android:anim/decelerate_interpolator：动画开始快然后慢。
- @android:anim/linear_interpolator：动画匀速改变。
- @android:anim/overshoot_interpolator：向前弹出一定值之后回到原来位置。

ANIMATION



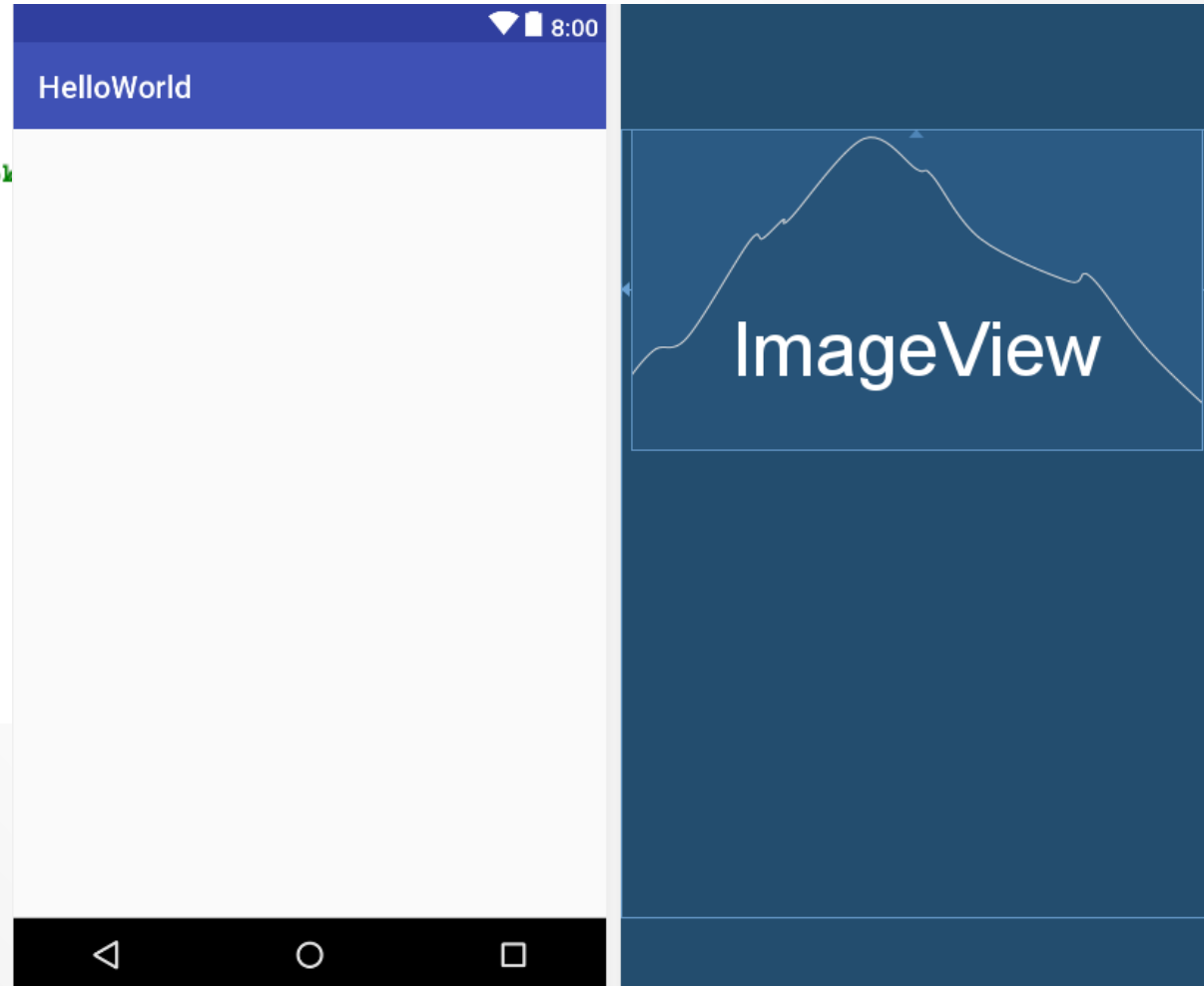
```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator">
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.0"
        android:startOffset="500"
        android:duration="2000"/>
    <rotate
        android:fromDegrees="0"
        android:toDegrees="+360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2000"/>
    <scale
        android:fromXScale="1.0"
        android:toXScale="0.0"
        android:fromYScale="1.0"
        android:toYScale="0.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2000"/>
    <translate
        android:fromXDelta="0%"
        android:toXDelta="100%"
        android:fromYDelta="0%"
        android:toYDelta="100%"
        android:duration="2000"/>
</set>
```

ANIMATION

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.bishop.helloworld.MainActivity">

    <ImageView
        android:id="@+id/animation"
        android:layout_width="370dp"
        android:layout_height="208dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



ANIMATION

```
package com.example.bishop.helloworld;

import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {
    ImageView img = null;
    Animation animation;
    AnimationDrawable draw;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onResume() {
        super.onResume();
        img = (ImageView) findViewById(R.id.animation);
        img.setBackgroundResource(R.drawable.animation);
        draw = (AnimationDrawable) MainActivity.this.img.getBackground();
        draw.start();
        animation = AnimationUtils.loadAnimation(context: MainActivity.this, R.anim.animation);
        img.startAnimation(animation);
    }
}
```



ANIMATION

- 逐帧动画 & 补间动画存在一定的缺点：
- 作用对象局限。有些情况下的动画效果只是视图的某个属性 & 对象而不是整个视图。
- 没有改变View的属性，只是改变视觉效果。补间动画只是改变了View的视觉效果，而不会真正去改变View的属性。
- 动画效果单一。补间动画只能实现平移、旋转、缩放 & 透明度这些简单的动画需求，一旦遇到相对复杂的动画效果，即超出了上述4种动画效果，那么补间动画则无法实现。

ANIMATION

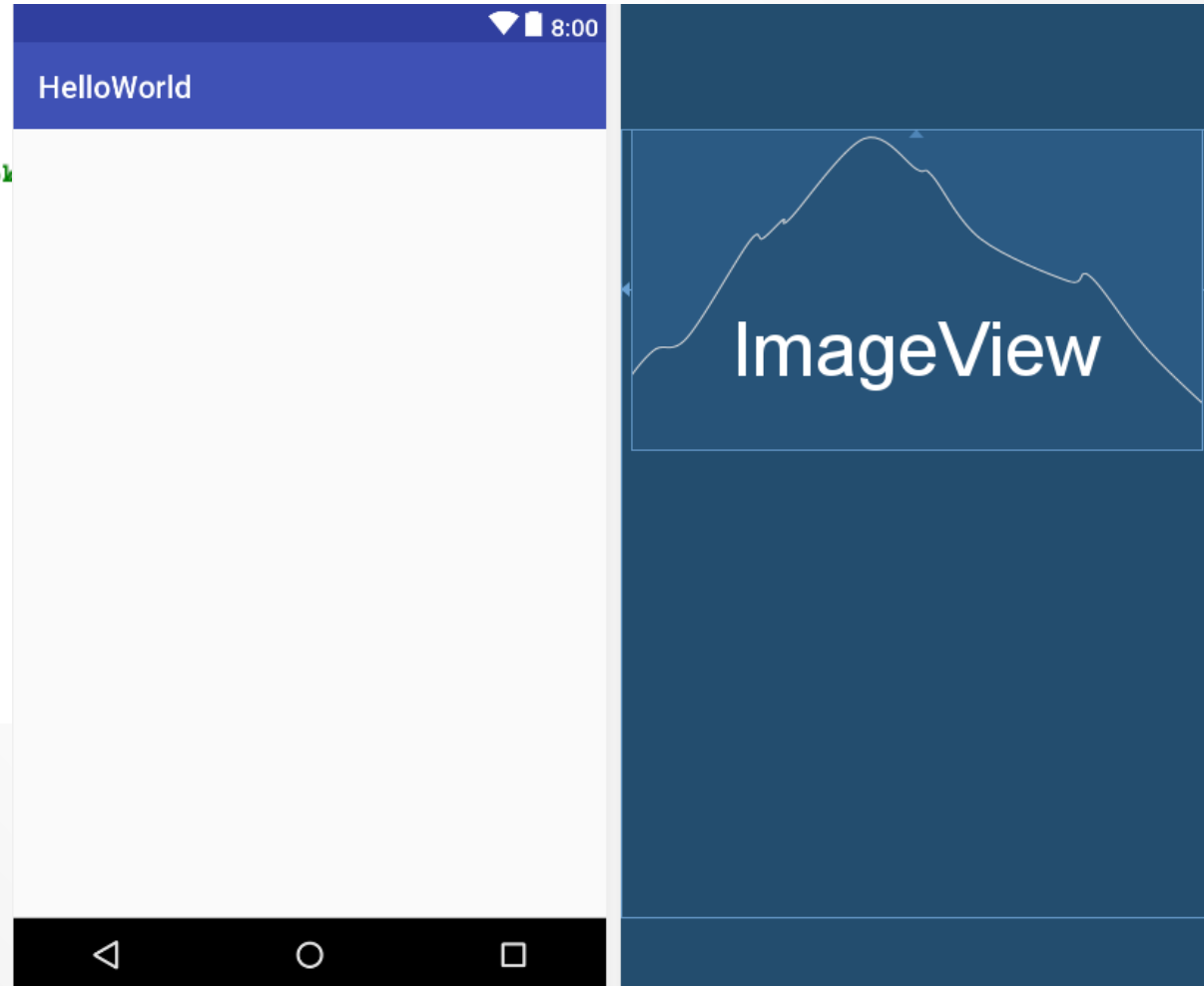
- 为了解决补间动画的缺陷，在 Android 3.0 (API 11) 开始，系统提供了一种全新的动画模式：属性动画 (Property Animation) 。
- 作用对象：任意 Java 对象，不再局限于 视图View对象。
- 实现的动画效果：可自定义各种动画效果，不再局限于4种基本变换：平移、旋转、缩放 & 透明度。
- 工作原理：在一定时间间隔内，通过不断对值进行改变，并不断将该值赋给对象的属性，从而实现该对象在该属性上的动画效果。

ANIMATION

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.bishop.helloworld.MainActivity">

    <ImageView
        android:id="@+id/animation"
        android:layout_width="370dp"
        android:layout_height="208dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



ANIMATION

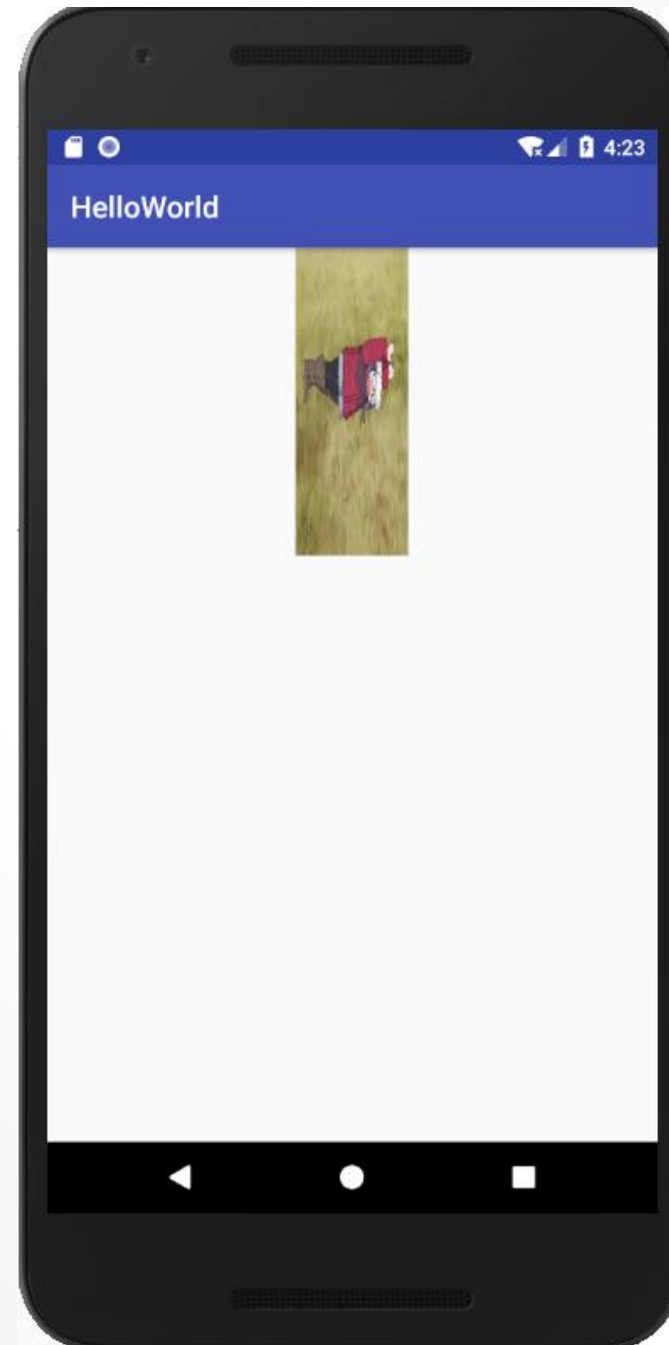
```
package com.example.bishop.helloworld;

import android.animation.Animator;
import android.animation.ValueAnimator;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ImageView;

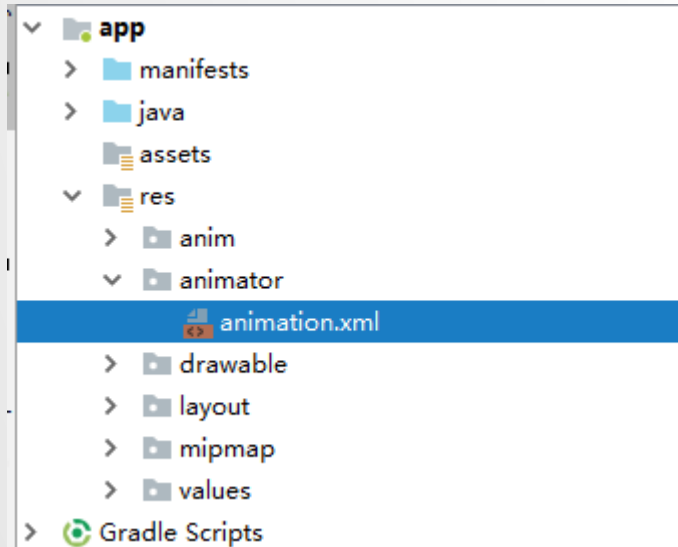
public class MainActivity extends AppCompatActivity {
    ImageView img = null;
    AnimationDrawable draw;
    Animator animator;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img = (ImageView) findViewById(R.id.animation);
        img.setBackgroundResource(R.drawable.animation);
        draw = (AnimationDrawable) MainActivity.this.img.getBackground();
        draw.start();
    }

    @Override
    protected void onResume() {
        super.onResume();
        ValueAnimator valueAnimator = ValueAnimator.ofInt(img.getLayoutParams().width, 200);
        valueAnimator.setDuration(2000);
        valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
            @Override
            public void onAnimationUpdate(ValueAnimator animator) {
                int currentValue = (Integer) animator.getAnimatedValue();
                System.out.println(currentValue);
                img.getLayoutParams().width = currentValue;
                img.requestLayout();
            }
        });
        valueAnimator.start();
    }
}
```



ANIMATION



```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:ordering="sequentially" >
    <objectAnimator
        android:duration="2000"
        android:propertyName="translationX"
        android:valueFrom="0"
        android:valueTo="300"
        android:valueType="floatType" >
    </objectAnimator>
    <objectAnimator
        android:duration="3000"
        android:propertyName="rotation"
        android:valueFrom="0"
        android:valueTo="360"
        android:valueType="floatType" >
    </objectAnimator>
</set>
```

ANIMATION

```
package com.example.bishop.helloworld;

import android.animation.AnimatorInflater;
import android.animation.AnimatorSet;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {
    ImageView img = null;
    AnimationDrawable draw;
    AnimatorSet animator;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img = (ImageView) findViewById(R.id.animation);
        img.setBackgroundResource(R.drawable.animation);
        draw = (AnimationDrawable) MainActivity.this.img.getBackground();
        draw.start();
        animator = (AnimatorSet) AnimatorInflater.loadAnimator(context, this, R.animator.animation);
        animator.setTarget(img);
        animator.start();
    }
}
```



ANIMATION LISTENER

- `anim.setAnimationListener(new AnimationListener() {`
- `@Override`
- `public void onAnimationCancel(Animation animation) {...}`
- `@Override`
- `public void onAnimationStart(Animation animation) {...}`
- `@Override`
- `public void onAnimationRepeat(Animation animation) {...}`
- `@Override`
- `public void onAnimationEnd(Animation animation) {...}`
- `});`

MEDIAPLAYER

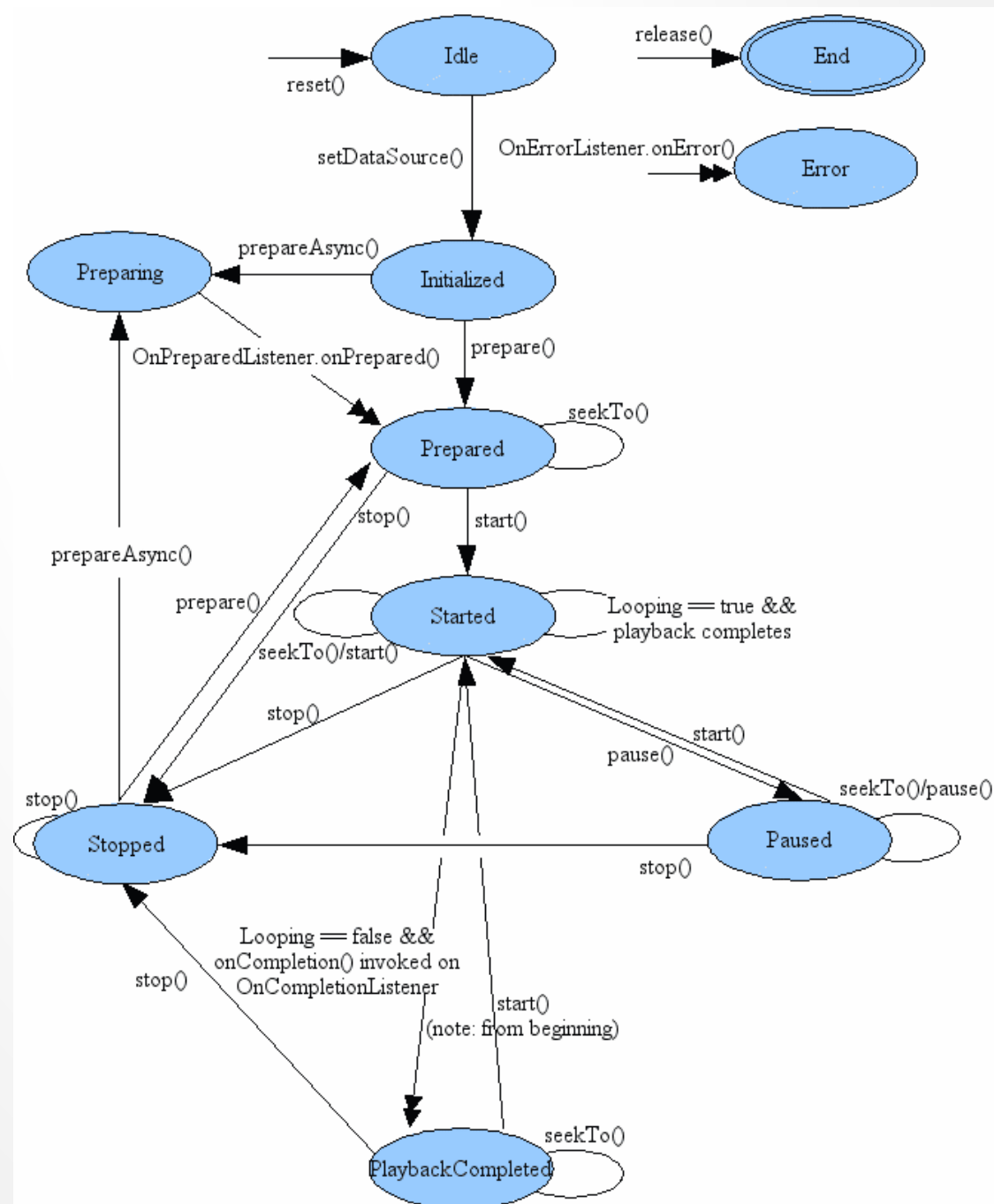
- Android可以通过MediaPlayer类提供的API，实现以下音频、视频文件的播放：
- 自带resource资源： `MediaPlayer.create(this, R.raw.test);`
- SD卡或其他文件路径下的媒体文件：
`mp.setDataSource("/sdcard/test.mp3");`
- 网络媒体文件：
`mp.setDataSource("http://www.citynorth.cn/music/confucius.mp3");`

MEDIAPLAYER

- setDataSource一共四个方法：
- setDataSource (String path) ; //文件系统路径
- setDataSource (FileDescriptor fd); //assets文件
- setDataSource (Context context, Uri uri); //网络虚拟路径
- setDataSource (FileDescriptor fd, long offset, long length);
- 若URI中包含网络资源，需在AndroidManifest.xml中申请Internet访问权限：
- `<uses-permission
android:name="android.permission.INTERNET" />`

MEDIAPLAYER

- MediaPlayer生命周期:



MEDIAPLAYER

- Idle 状态：当使用new()方法创建一个MediaPlayer对象或者调用了其reset()方法时，该MediaPlayer对象处于idle状态。这两种方法的一个重要差别就是：如果在这个状态下调用了getDuration()等方法（相当于调用时机不正确），通过reset()方法进入idle状态的话会触发OnErrorListener.onError()，并且MediaPlayer会进入Error状态；如果是新创建的MediaPlayer对象，则并不会触发onError(),也不会进入Error状态。
- End 状态：通过release()方法可以进入End状态，只要MediaPlayer对象不再被使用，就应当尽快将其通过release()方法释放掉，以释放相关的软硬件组件资源，这其中有些资源是只有一份的（相当于临界资源）。如果MediaPlayer对象进入了End状态，则不会在进入任何其他状态了。
- Initialized 状态：这个状态比较简单，MediaPlayer调用setDataSource()方法就进入Initialized状态，表示此时要播放的文件已经设置好了。

MEDIAPLAYER

- Prepared 状态：初始化完成之后还需要通过调用prepare()或prepareAsync()方法，这两个方法一个是同步的一个是异步的，只有进入Prepared状态，才表明MediaPlayer到目前为止都没有错误，可以进行文件播放。
- Preparing 状态：这个状态比较好理解，主要是和prepareAsync()配合，如果异步准备完成，会触发OnPreparedListener.onPrepared()，进而进入Prepared状态。MediaPlayer准备资源调用prepare()时，会执行一段稍长的时间，因为它在解码媒体数据，如果解码时间过长那么会出现主线程阻塞，从而触发ANR异常，导致程序运行很慢，所以框架提供了prepareAsync()异步准备方法并提供资源准备监听，当资源准备完成会触发MediaPlayer.OnPreparedListener的onPrepared()方法。

MEDIAPLAYER

- Started 状态：显然，MediaPlayer一旦准备好，就可以调用start()方法，这样MediaPlayer就处于Started状态，这表明MediaPlayer正在播放文件过程中。可以使用isPlaying()测试MediaPlayer是否处于了Started状态。如果播放完毕，而又设置了循环播放，则MediaPlayer仍然会处于Started状态，类似的，如果在该状态下MediaPlayer调用了seekTo()或者start()方法均可以让MediaPlayer停留在Started状态。
- Paused 状态：Started状态下MediaPlayer调用pause()方法可以暂停MediaPlayer，从而进入Paused状态，MediaPlayer暂停后再次调用start()则可以继续MediaPlayer的播放，转到Started状态，暂停状态时可以调用seekTo()方法，这是不会改变状态的。

MEDIAPLAYER

- Stop 状态: Started或者Paused状态下均可调用stop()停止MediaPlayer, 而处于Stop状态的MediaPlayer要想重新播放, 需要通过prepareAsync()和prepare()回到先前的Prepared状态重新开始才可以。
- PlaybackCompleted状态: 文件正常播放完毕, 而又没有设置循环播放的话就进入该状态, 并会触发OnCompletionListener的onCompletion()方法。此时可以调用start()方法重新从头播放文件, 也可以stop()停止MediaPlayer, 或者也可以seekTo()来重新定位播放位置。

MEDIAPLAYER

- Error状态：如果由于某种原因MediaPlayer出现了错误，会触发OnErrorListener.onError()事件，此时MediaPlayer即进入Error状态，及时捕捉并妥善处理这些错误是很重要的，可以帮助我们及时释放相关的软硬件资源，也可以改善用户体验。
- 通过
setOnErrorListener(android.media.MediaPlayer.OnErrorListener)
可以设置该监听器。如果MediaPlayer进入了Error状态，可以通过调用reset()来恢复，使得MediaPlayer重新返回到Idle状态。

MEDIAPLAYER

- 常用方法：
- `int getCurrentPosition()`: 获取当前播放的位置。
- `int getAudioSessionId()`: 返回音频的session ID。
- `int getDuration()`: 得到文件的时间。
- `TrackInfo[] getTrackInfo()`: 返回一个track信息的数组。
- `boolean isLooping ()`: 是否循环播放。
- `boolean isPlaying()`: 是否正在播放。

MEDIAPLAYER

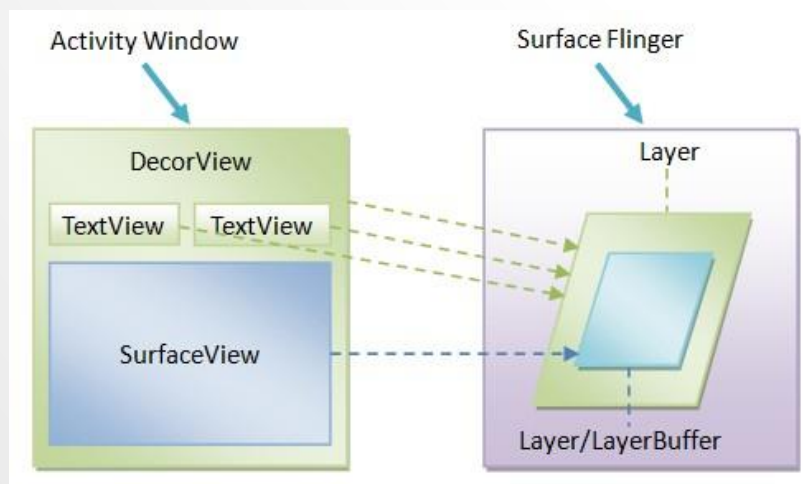
- void pause (): 暂停。
- void start (): 开始。
- void stop (): 停止。
- void prepare(): 同步的方式装载流媒体文件。
- void prepareAsync(): 异步的方式装载流媒体文件。
- void reset(): 重置MediaPlayer至未初始化状态。
- void release (): 回收流媒体资源。
- void seekTo(int msec): 指定播放的位置（以毫秒为单位的时间）。

MEDIAPLAYER

- `void setAudioStreamType(int streamtype)`: 指定流媒体类型。
- `void setLooping(boolean looping)`: 设置是否单曲循环。
- `void setNextMediaPlayer(MediaPlayer next)`: 当这个MediaPlayer播放完毕后, MediaPlayer next开始播放。
- `void setWakeMode(Context context, int mode)`: 设置CPU唤醒的状态。
- `void setScreenOnWhilePlaying(Boolean screenOn)`: 播放时是否保持屏幕常亮, 是否阻止屏幕自动休眠。

MEDIAPLAYER

- View通过刷新来重绘视图，Android系统通过发出VSYNC信号来进行屏幕的重绘，刷新的时间间隔为16ms。在一些需要频繁刷新，执行很多逻辑操作的时候，超过了16ms，就会导致卡顿。
- SurfaceView继承自View，但拥有独立的绘制表面，即它不与其宿主窗口共享同一个绘图表面，可以单独在一个线程进行绘制，并不会占用主线程的资源。这样，绘制就会比较高效，游戏，视频播放，还有最近热门的直播，都可以用SurfaceView。



MEDIAPLAYER

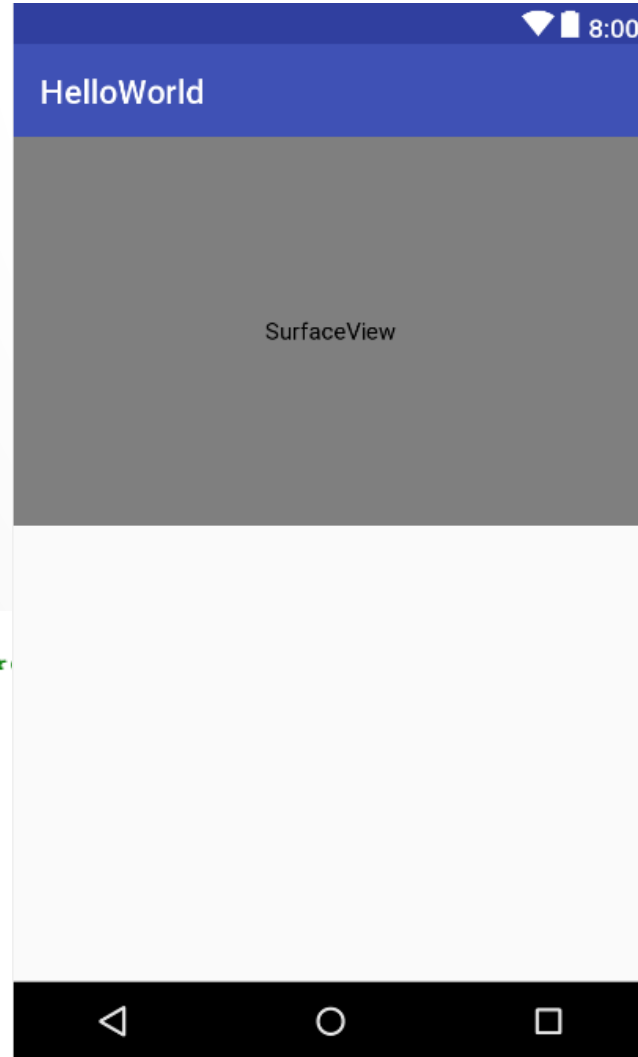
- SurfaceView和View的区别：
- View主要适用于主动更新的情况下，而SurfaceView主要适用于被动更新，例如频繁地刷新。
- View在主线程中对画面进行刷新，而SurfaceView通常会通过一个子线程来进行页面的刷新。
- View在绘图时没有使用双缓冲机制，而SurfaceView在底层实现机制中就已经实现了双缓冲机制。

MEDIAPLAYER

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld"
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="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>

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.bishop.helloworld.MainActivity">

    <SurfaceView
        android:id="@+id/surfaceView"
        android:layout_width="match_parent"
        android:layout_height="235dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```



MEDIAPLAYER

```
package com.example.bishop.helloworld;

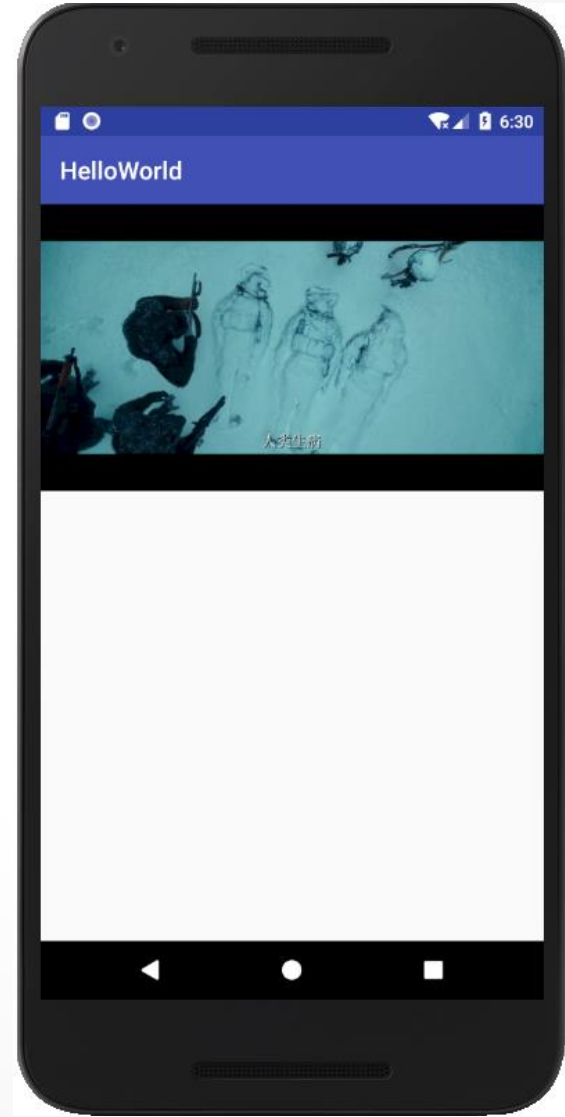
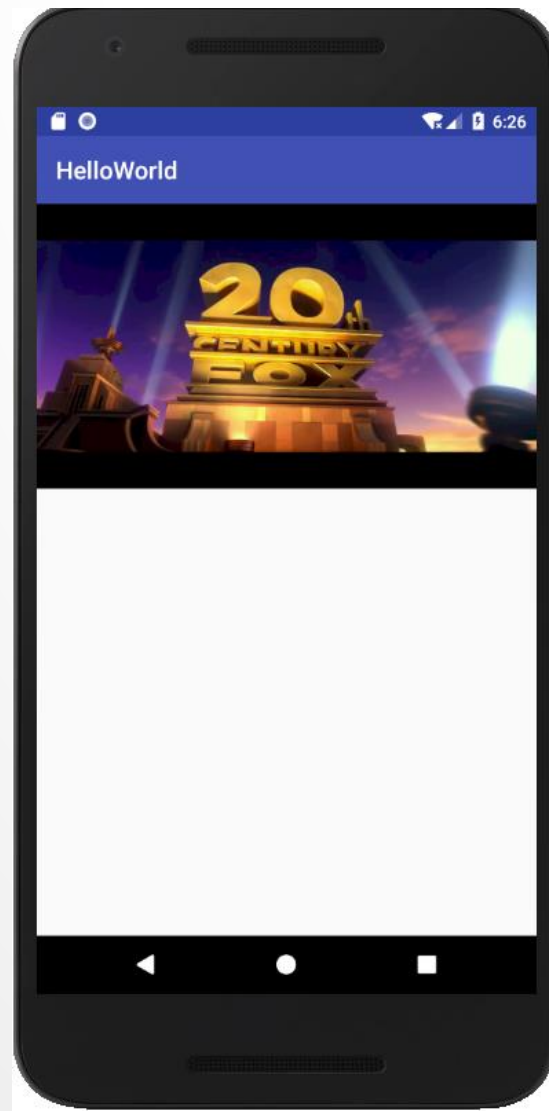
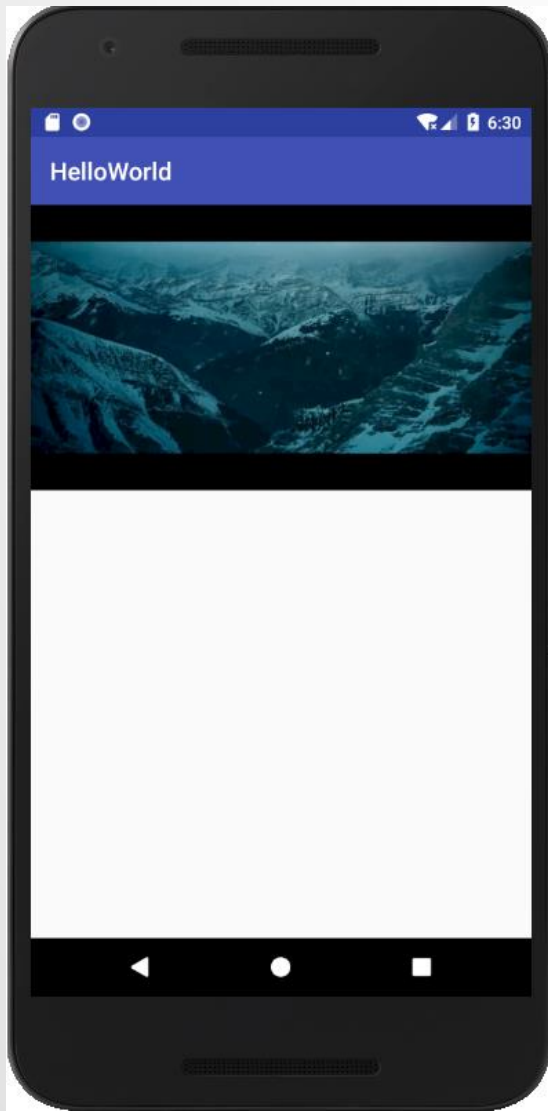
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.SurfaceView;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {
    MediaPlayer mp;
    SurfaceView mySurface;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mySurface=(SurfaceView)findViewById(R.id.surfaceView);
        mp=new MediaPlayer();
        Uri uri=Uri.parse("http://vfx.mtime.cn/Video/2017/03/31/mp4/170331093811717750.mp4");
        mp.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mp) {
                mp.setDisplay(mySurface.getHolder());
                mp.start();
            }
        });
    }
}
```

```
try {
    mp.setDataSource(context, MainActivity.this, uri);
    mp.prepareAsync();
} catch (IOException e) {
    e.printStackTrace();
}

@Override
protected void onDestroy(){
    super.onDestroy();
    if(mp.isPlaying()){
        mp.stop();
    }
    mp.release();
}
```

MEDIAPLAYER



THE END.