西南交通大學
Southwest Jiaotong University

移动商务应用开发

第12章 多媒体应用

学习内容

主要内容

12.1 图像与绘图

12.2 动画

12.3 音频播放

12.4 视频播放

图像与绘图：Bitmap

- Bitmap指的是位图，位图是由像素点组成的。

图像与绘图：Bitmap

- Bitmap指的是位图，位图是由像素点组成的。
  - 位深度
  - 分辨率

图像与绘图：Bitmap

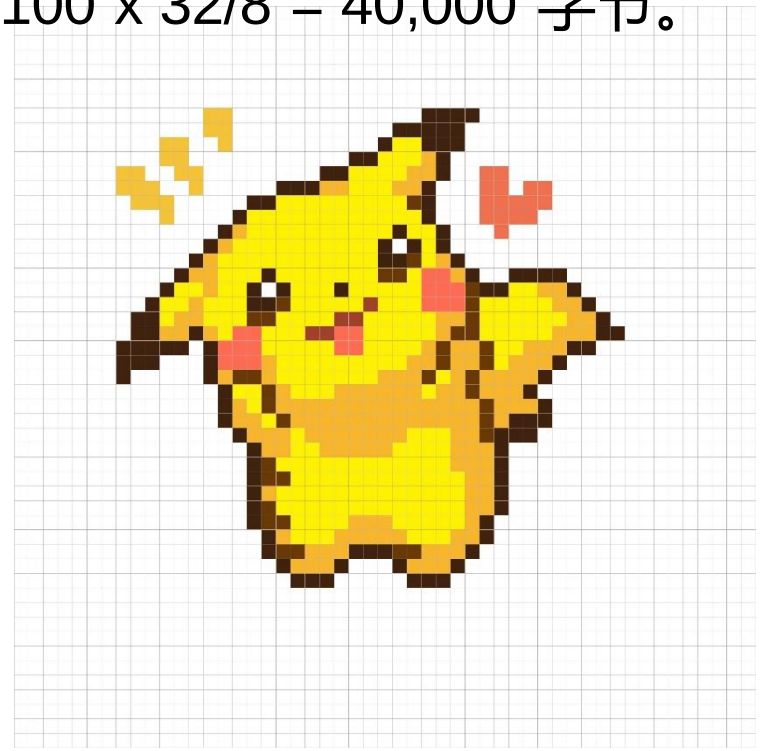
- 位深度指的是每个像素点能表示的颜色数量。
- 位深度通常包括1、8、16、24、32位色彩。位深度越大，每个像素点能表示的颜色越多

图像与绘图：Bitmap

- 分辨率指的是图像中包含的像素的数量，通常以宽度和高度的像素数量表示。
- 如果一张图像的分辨率为100 x 100，那表明该图像在水平方向上有100个像素点，在垂直方向上具有100个像素点。

图像与绘图：Bitmap

- 图片的大小取决于图片的分辨率与位深度。
- 如果一张图像的分辨率为100 x 100，位深度为32，那么该图片占用 100 x 100 x 32/8 = 40,000 字节。



图像与绘图：Bitmap

- Android 提供了Bitmap类和BitmapFactory类来操作位图。
- Bitmap类提供了一些方法，包括静态方法和成员方法。
  - 静态方法由 static 关键字声明，可以直接通过类名调用
  - 成员方法无 static 关键字，只能通过对象调用。

图像与绘图：Bitmap

- Bitmap类的部分静态方法

方法	说明
Bitmap **createBitmap**(Bitmap src)	复制位图
Bitmap **createBitmap**(int width, int height, Config config)	创建一个指定宽、高、色彩深度的新位图
Bitmap **createBitmap**(Bitmap src, int x, int y, int width, int height)	从原位图src的坐标(x,y)开始截取一个指定宽、高的部分，用于创建新位图
Bitmap **createScaledBitmap**(Bitmap src, int width, int height, boolean filter)	将从原位图src缩放成一个指定宽、高的新位图

图像与绘图：Bitmap

- Bitmap类的部分静态方法（补充）
  - Bitmap.Config为Bitmap类内部的枚举类，提供了不同的位深度

成员	说明
ALPHA_8	8位
ARGB_4444	16位
ARGB_8888	32位（默认）

  - filter用于指定在缩放过程中使用的过滤方法。如果为true，则使用双线性过滤，如果为false，则使用最近过滤方法。

图像与绘图：Bitmap

- Bitmap类的部分成员方法

方法	说明
int getWidth()	获取位图的宽度
int getHeight()	获取位图的高度
Config getConfig()	获取位图的位深度
compress(Bitmap.CompressFormat format, int quality, OutputStream stream)	将位图压缩并写入到输出流中
void recycle()	强制回收位图资源
boolean isRecycled()	判断位图内存是否已释放

图像与绘图：Bitmap

- Bitmap类的部分成员方法（补充）
  - compress()通常用于将位图保存为本地文件或者发送位图数据。
  - Bitmap.CompressFormat为Bitmap类的内部枚举类，提供了不同的压缩格式
    - Bitmap.CompressFormat.JPEG
    - Bitmap.CompressFormat.PNG
    - Bitmap.CompressFormat.WEBP
  - quality表示压缩后的质量，取值范围为0到100，值越大表示压缩后质量越高。

图像与绘图：Bitmap

- 拓展：如何将Bitmap对象保存成本地文件
- 外部存储

```
File file = new File(fileName);
FileOutputStream fos = new FileOutputStream(file);
bitmap.compress(Bitmap.CompressFormat.PNG, 100, fos);
```

- 内部存储

```
FileOutputStream fos = context.openFileOutput(fileName, Context.MODE_PRIVATE);
bitmap.compress(Bitmap.CompressFormat.PNG, 100, fos);
```

图像与绘图：Bitmap

- 拓展：如何将Bitmap对象保存成本地文件
- 保存到相册 (相册是个数据库，利用ContentResolver)

```
ContentValues values = new ContentValues();
// 插入空的内容，获得一个插入后的URI
Uri uri = getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values);
if (uri != null) {
    try {
        // 获取指定URI的输出流
        OutputStream os = getContentResolver().openOutputStream(uri);
        // 将图片压缩到指定输出流
        bitmap.compress(Bitmap.CompressFormat.JPEG, 100, os);
        os.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

图像与绘图：Bitmap

- Bitmap类的部分成员方法（补充）
  - 由于Bitmap是高质量的图片格式，需要占用较大存储空间，所以当不再需要Bitmap对象时，为了及时释放内存，应当判断Bitmap对象是否已被回收，如果没有则回收。

```
protected void onDestroy() {
    super.onDestroy();
    if (!bitmap.isRecycled()) {
        bitmap.recycle();
    }
}
```

图像与绘图：Bitmap

- BitmapFactory类是一个用于创建Bitmap对象的工具类，提供了多种方法用于从不同的数据源中解码得到Bitmap对象。
  - 不同的数据源：字节数组、文件、资源、输入流等

方法	说明
Bitmap **decodeByteArray**(byte[] data, int offset, int length)	从字节数组中解码生成一个位图
Bitmap **decodeFile**(String pathName)	从文件中解码生成一个位图
Bitmap **decodeResource**(Resources res, int id)	根据指定的资源id，从资源中解码位图
Bitmap **decodeStream**(InputStream is)	从输入流中解析出位图

图像与绘图：Bitmap

- 示例：BitmapFactory类使用

```
// 从字节数组中解码位图
byte[] data = response.body().bytes();
bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
// 从文件中解码位图
String filePath = "/sdcard/pictures/example.jpg";
bitmap = BitmapFactory.decodeFile(filePath);
// 从资源中解码位图
bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.background);
// 从输入流中解码位图
FileInputStream in = new FileInputStream("/sdcard/pictures/example.jpg");
bitmap = BitmapFactory.decodeStream(in);
```

图像与绘图：Paint

- Android绘图主要依靠以下几个类：
  - Paint：画笔，用于设置画笔的属性。
  - Canvas：画布，用于绘制各种形状。
  - Bitmap：位图，用于保存绘画。
- Paint即画笔。通过Paint类可以创建画笔，并设置画笔的线宽、颜色、透明度和填充风格等属性。
  - 创建画笔

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

  - 设置画笔的各种属性

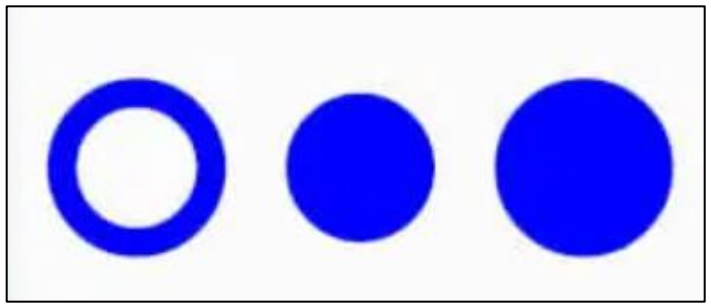
图像与绘图：Paint

- 设置画笔的各种属性

方法	说明
setARGB(int a,int r,int g,int b)	设置画笔的透明度和颜色，a表示透明度，r、g、b 表示颜色值。
setColor(int color)	设置画笔的透明度和颜色
setAntAlias(boolean aa)	设置是否抗锯齿（抗锯齿会更平滑）
setDither(boolean dither)	设置是否放抖动（防抖动会更平滑）
setStyle(Paint.Style style)	设置画笔的样式，如FILL、STROKE、FILL_AND_STROKE
setStrokeWidth(float width)	设置画笔的线条粗细
setStrokeCap(Paint.Cap cap)	设置线条端点的形状，如圆形、方形等
setStrokeJoin(Paint.Join join)	设置线条连接处的形状，如圆角、直角等
setPathEffect(PathEffect effect)	设置绘制路径的效果，如虚线、点画线等

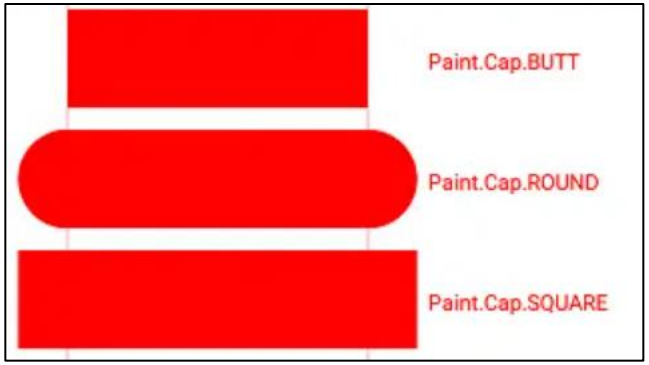
图像与绘图：Paint

- 设置画笔的各种属性（补充）
  - ARGB取值范围为 0-255。
  - Paint.Style是Paint类的内部枚举类，提供了不同样式，包括：
    - STROKE（描边）
    - FILL（填充）
    - FILL_AND_STROKE（填充且描边）



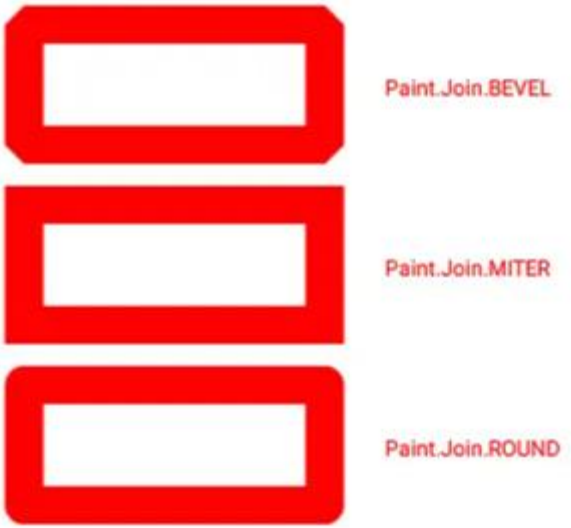
图像与绘图：Paint

- 设置画笔的各种属性（补充）
  - Paint.Cap提供了不同的线条端点样式，包括：
    - Cap.BUTT （无线帽）
    - Cap.ROUND （圆形线帽）
    - Cap.SQUARE （方形线帽）



图像与绘图：Paint

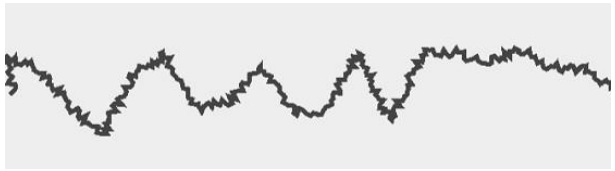
- 设置画笔的各种属性（补充）
  - Paint.Join提供了不同的线条连接处的样式，包括：
    - Join.BEVEL（结合处为锐角）
    - Join.MITER（结合处为直线）
    - Join.Round（结合处为圆弧）



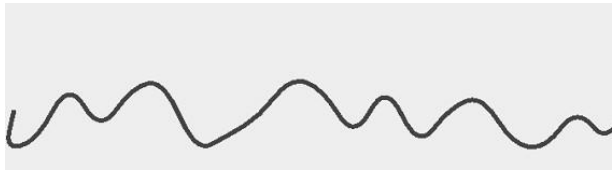
图像与绘图：Paint

- 设置画笔的各种属性（补充）
  - PathEffect提供了不同的路径效果，包括：
    - CornerPathEffect（将路径的转角变得圆滑）
    - DiscretePathEffect（模拟出类似生锈铁丝的效果）
    - DashPathEffect（虚线）

没有路径效果



DiscretePathEffect



CornerPathEffect

DashPathEffect

图像与绘图：Canvas

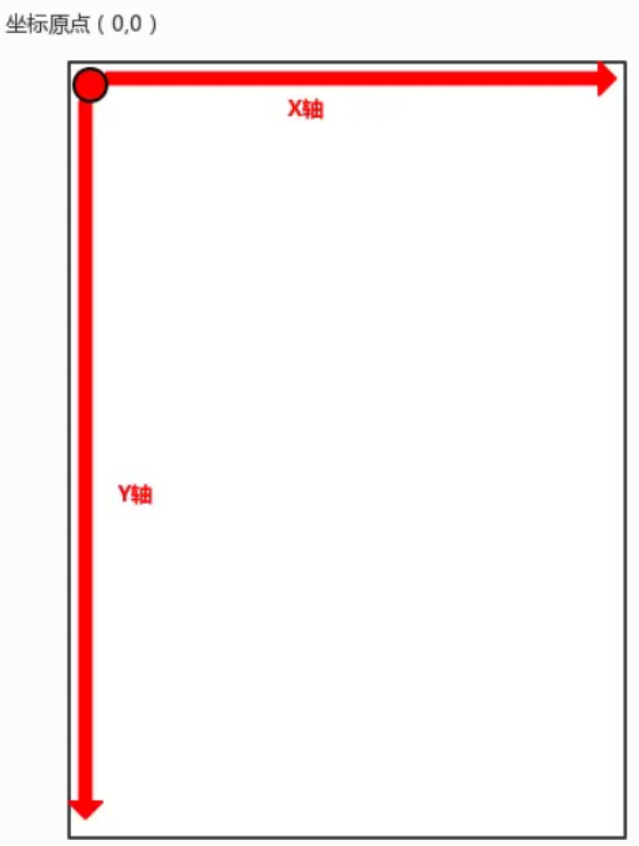
- Canvas即画布。通过Canvas类可以绘制各种形状、文本、图像、路径。
- 创建Canvas对象
  - 用Bitmap对象创建Canvas对象，绘画的内容将会被保存到Bitmap对象中。

```
// 创建一个空白的Bitmap
Bitmap bitmap = Bitmap.createBitmap(400, 400, Bitmap.Config.ARGB_8888);
// 用 Bitmap创建Canvas对象
Canvas canvas = new Canvas(bitmap);
```

  - 之后可以将Bitmap显示在ImageView中，保存到本地。

图像与绘图：Canvas

- Canvas坐标系
  - 默认的坐标系：原点在左上角



图像与绘图：Canvas

- Canvas坐标系
  - 默认的坐标系：原点在左上角
  - 可以调整坐标系

方法	说明
translate(float dx, float dy)	平移坐标系
rotate(float degrees)	旋转坐标系
scale(float sx, float sy)	缩放坐标系
skew(float sx, float sy)	倾斜坐标系

图像与绘图：Canvas

- Canvas绘制形状、文本、图像、路径

方法	说明
drawLine(float startX, float startY, float stopX, float stopY, Paint paint)	根据起点和终点绘制一条直线
drawRect(RectF rect, Paint paint)	绘制一个矩形
drawRoundRect(RectF rect, float rx, float ry, Paint paint)	根据矩形和圆角半径绘制一个圆角矩形
drawCircle(float cx, float cy, float radius,Paint paint)	根据圆点和半径绘制一个圆
drawOval(RectF oval, Paint paint)	根据矩形，绘制一个椭圆
drawArc(RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint)	根据矩形、起点和终点的角度，绘制一个弧形或扇形
drawPoint(float x, float y, Paint paint)	根据坐标，绘制一个点
drawText(String text, float x, floaty, Paint paint)	绘制文本
drawBitmap(Bitmap bitmap, float left, float top, Paint paint)	绘制图像
drawPath(Path path, Paint paint)	绘制一条路径

图像与绘图：Canvas

- Canvas绘制形状、文本、图像、路径（补充）
  - RectF类表示矩形，包括 (float left, float top, float right, float bottom) 四个属性。
  - Path类表示路径。
    - 创建Path对象
      - Path path = new Path();
    - Path类的主要方法

图像与绘图：Canvas

- Canvas绘制形状、文本、图像、路径（补充）
  - Path类的主要方法

方法	说明
moveTo(float x, float y)	移动到指定的坐标位置
lineTo(float x, float y)	绘制一条到指定位置的直线
quadTo(float x1, float y1, float x2, float y2)	绘制一条二次贝塞尔曲线
cubicTo(float x1, float y1, float x2, float y2, float x3, float y3)	绘制一条三次贝塞尔曲线
arcTo(RectF oval, float startAngle, float sweepAngle, boolean forceMoveTo)	绘制一个弧线
addCircle(float x, float y, float radius, Path.Direction dir)	添加一个圆到路径中
addRect(RectF rect, Path.Direction dir)	添加一个矩形到路径中
close()	闭合路径

图像与绘图：自定义View

- 拓展：自定义View（简介）
- 通过自定义View可以自己创建和控制UI的外观和行为。
- 如何自定义View?
  1. 创建一个继承View类的Java类
  2. 重写构造方法：至少需要重写**两个**构造方法
     - View(Context context)：在**Java代码中**直接创建View对象时使用
     - View(Context context, AttributeSet attrs)：在**XML布局文件中**使用，attrs包含了该View的属性
  3. 重写onDraw()方法：传入一个Canvas对象，用于绘制View的外观。

图像与绘图：自定义View

- 拓展：自定义View（简介）
- View的部分方法

方法	说明
invalidate()	请求重新绘制，会再次调用onDraw()
onTouchEvent(MotionEvent event)	处理触摸事件
getWidth()	获取视图的宽度
getHeight()	获取视图的高度

图像与绘图：自定义View

- 拓展：ShapeableImageView（简介）
- 通过ShapeableImageView可以自定义ImageView的样式
- 如何使用ShapeableImageView?

1. 添加依赖

   implementation("com.google.android.material:material:1.12.0")

2. 在res/values/styles.xml 中定义使用的样式

```
<!-- 圆形 -->
<style name="circleImageStyle">
    <item name="cornerFamily">rounded</item>
    <item name="cornerSize">50%</item>
</style>
```

图像与绘图：自定义View

- 拓展：ShapeableImageView（简介）
- 如何使用ShapeableImageView?
  1. 添加依赖
  2. 在res/values/styles.xml 中定义使用的样式
  3. 在布局文件中添加ShapeableImageView，并设置其属性 app:shapeAppearanceOverlay

```
<com.google.android.material.imageview.ShapeableImageView
    android:id="@+id/imageViewPrevious"
    android:layout_width="60dp"
    android:layout_height="60dp"
    app:shapeAppearanceOverlay="@style/circleImageStyle"
    app:srcCompat="@drawable/baseline_skip_previous_24"/>
```

图像与绘图：自定义View

- 拓展：ShapeableImageView（简介）
- 不同的样式

```
<!-- 圆形 -->
<style name="circleImageStyle">
    <item name="cornerFamily">rounded</item>
    <item name="cornerSize">50%</item>
</style>
```


```
<!--圆角-->
<style name="RoundedCornerImageStyle">
    <item name="cornerFamily">rounded</item>
    <item name="cornerSize">10dp</item>
</style>
```

图像与绘图：自定义View

- 拓展：ShapeableImageView（简介）
- 不同的样式

```
<!-- 切角图片 -->
<style name="cutImageStyle">
    <item name="cornerFamily">cut</item>
    <item name="cornerSize">12dp</item>
</style>
```


```
<!-- 菱形图片 -->
<style name="diamondImageStyle">
    <item name="cornerFamily">cut</item>
    <item name="cornerSize">50%</item>
</style>
```

学习内容

12.1 图像与绘图

12.2 动画

12.3 音频播放

12.4 视频播放

主要内容

动画：逐帧动画

- 逐帧动画是一张张地播放事先准备好的静态图像
- 逐帧动画是通过AnimationDrawable类来实现的。
- 可以通过XML创建逐帧动画，也可以用Java代码创建逐帧动画。

动画：逐帧动画

- 通过XML创建逐帧动画

1. 在res/drawable下创建 Root name = animation-list 的xml文件

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/song1" android:duration="300"/>
    <item android:drawable="@drawable/song2" android:duration="300"/>
    <item android:drawable="@drawable/song3" android:duration="300"/>
</animation-list>
```

2. 在布局文件中添加一个 ImageView 来显示动画

3.在Java代码中控制动画

```
ImageView imageView = findViewById(R.id.imageView);
imageView.setImageResource(R.drawable.frame_animation);
AnimationDrawable animation = (AnimationDrawable) imageView.getDrawable();
animation.start();
```

动画：逐帧动画

- 通过Java代码创建逐帧动画

```
// 创建 AnimationDrawable 对象
AnimationDrawable animationDrawable = new AnimationDrawable();
// 添加帧到 AnimationDrawable
animationDrawable.addFrame(getDrawable(R.drawable.song1), 300);
animationDrawable.addFrame(getDrawable(R.drawable.song2), 300);
animationDrawable.addFrame(getDrawable(R.drawable.song3), 300);
// 将 AnimationDrawable 设置为 ImageView 的显示内容
imageView.setImageDrawable(animationDrawable);
// 启动动画
animationDrawable.start();
```

动画：逐帧动画

- AnimationDrawable类的主要方法：

方法	说明
start()	启动动画
stop()	停止动画
isRunning()	检查动画是否正在运行
addFrame(Drawable frame, int duration)	添加一帧到动画中，frame表示静态图像，duration表示持续时间
getNumberOfFrames()	返回动画中的帧数
getFrame(int index)	获取指定索引的帧
getDuration(int index)	获取指定索引的帧的持续时间

动画：补间动画

- 补间动画只需要设置动画开始、动画结束等关键帧，动画变化的中间帧由系统计算并补全，可以实现平移、缩放、旋转、改变透明度。
- 补间动画是通过Animation类来实现的。
- 可以通过XML创建补间动画，也可以用Java代码创建补间动画。

动画：补间动画

- 通过XML创建补间动画（以平移为例）

1. 创建resource type = anim的资源文件夹，并创建一个Root element = set的xml文件

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:fromYDelta="-100%" // 表示相较于自己的大小
        android:toXDelta="0"
        android:toYDelta="100%"
        android:duration = "1000"
        android:repeatCount = "1"/>
</set>
```

2. 在Java代码中控制动画

动画：补间动画

- 通过XML创建补间动画（以平移为例）
  1. 创建resource type = anim的资源文件夹，并创建一个Root element = set的xml文件
  2. 在Java代码中控制动画

```
imageView.setImageResource(R.mipmap.ic_launcher_round);
// 加载动画
Animation animation = AnimationUtils.loadAnimation(this, R.anim.translate_animation);
// 启动动画
imageView.startAnimation(animation);
```

动画：补间动画

通过XML创建补间动画

- 缩放

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:toXScale="0.2"
        android:toYScale="0.2"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration = "1000"
        android:repeatCount = "1"/>
</set>
```

动画：补间动画

通过XML创建补间动画

- 旋转

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="1000"
        android:repeatCount = "1"/>
</set>
```

动画：补间动画

通过XML创建补间动画

- 改变透明度

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.0"
        android:duration="1000"
        android:repeatCount = "1"/>
</set>
```

动画：补间动画

- 通过Java文件创建补间动画（以旋转为例）

```
// 通过Java代码创建补间动画（RotateAnimation）
RotateAnimation animation = new RotateAnimation(0f, 360f,
Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
animation.setDuration(1000); // 设置持续时间
animation.setRepeatCount(1); // 重复1次，共2次
imageView.startAnimation(animation); // 启动动画
```

动画：补间动画

- 通过Java文件创建补间动画

平移动画（TranslateAnimation）

旋转动画（RotateAnimation）

缩放动画（ScaleAnimation）

改变透明度动画（AlphaAnimation）

动画：补间动画

- Animation类的一些方法：
  - 补间动画本身没有提供暂停播放、继续播放的方法，但也可以通过一些技巧来实现这些效果。

方法	说明
start()	启动动画
cancel()	取消动画，动画将立即停止
getDuration()	获得总持续事件
getCurrentPlayTime()	获得当前播放进度
setDuration(long durationMillis)	设置动画持续时间
setInterpolator(TimeInterpolator value)	设置动画的插值器，用于控制动画的速度变化
setRepeatCount(int repeatCount)	设置动画重复次数
setRepeatMode(int repeatMode)	设置动画重复模式，正向或反向

动画：属性动画

- 属性动画比补间动画更灵活，可以实现平移、缩放、旋转、改变透明度等功能。
- 属性动画是通过Animator类或其子类ObjectAnimator类来实现的。
- 属性动画提供了暂停播放、继续播放的方法。
- 可以通过XML创建属性动画，也可以用Java代码创建属性动画。

动画：属性动画

- 通过XML创建属性动画（以旋转为例）

1. 创建resource type = animator的资源文件夹，并创建一个Root element = set的xml文件

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:propertyName= "rotation"
        android:valueFrom="0"
        android:valueTo="360"
        android:valueType="floatType"
        android:duration="1000"
        android:interpolator="@android:anim/linear_interpolator"
        android:repeatCount = "1"/>
</set>
```

2. 在Java代码中控制动画

动画：属性动画

通过XML创建属性动画

1. 创建resource type = animator的资源文件夹，并创建一个Root element = set的xml文件
2. 在Java代码中控制动画

```
imageView.setImageResource(R.mipmap.ic_launcher_round);
// 加载动画
Animator animator = AnimatorInflater.loadAnimator(this, R.animator.rotate_animator);
animator.setTarget(imageView);
// 启动动画
animator.start();
```

动画：属性动画

- 通过Java文件创建属性动画

```
ObjectAnimator animator = ObjectAnimator.ofFloat(imageView, "rotation",
0f, 360.0f); // 创建 ObjectAnimator 对象
animator.setDuration(1000); // // 设置动画持续时间
animator.setInterpolator(new LinearInterpolator()); // 匀速
animator.setRepeatCount(Animation.INFINITE); // -1 表示设置动画无限循环
animator.start(); // 启动动画
animator.pause(); // 暂停动画
animator.resume(); // 继续播放动画
animator.end(); // 结束动画
```

动画：属性动画

- ObjectAnimator类的一些方法

方法	说明
ofFloat(Object target, String propertyName, float… values)	创建 ObjectAnimator 对象
ofInt(Object target, String propertyName, int… values)	创建 ObjectAnimator 对象
ofArgb(Object target, String propertyName, int… values)	创建 ObjectAnimator 对象
setDuration(long duration)	设置动画的持续时间
setInterpolator(TimeInterpolator value)	设置动画的插值器，用于控制动画的速度变化
setRepeatCount(int value)	设置动画重复的次数
setRepeatMode(int value)	设置动画重复的模式，正向或反向
start()	启动动画
pause()	暂停动画。暂停后，动画停留在当前位置
resume()	恢复动画。恢复后，动画继续播放
cancel()	取消动画。取消后，动画停留在当前位置
end()	结束后，动画直接跳转到结束状态

动画：属性动画

- ObjectAnimator类的一些方法（补充）
  - propertyName主要包括：

选项	说明
translationX	在 X 轴上的平移距离
translationY	在 Y 轴上的平移距离
scaleX	在 X 轴上的缩放比例
scaleY	在 Y 轴上的缩放比例
rotation	围绕其中心点的旋转角度
rotationX	围绕 X 轴旋转的角度
rotationY	围绕 Y 轴旋转的角度
alpha	透明度

学习内容

12.1 图像与绘图

12.2 动画

12.3 音频播放

12.4 视频播放

主要内容

音频播放：MediaPlayer

- Android中播放音频有两种方式：
  - 采用MediaPlayer播放较长但对反应速度要求不高的音频文件，如音乐。
  - 采用SoundPool播放短暂但对反映速度要求较高的音频文件，如音效。

音频播放：MediaPlayer

- 通过MediaPlayer类可以播放音乐，并实现开始播放、暂停播放或停止播放等功能。

音频播放：MediaPlayer

- 创建MediaPlayer对象

方法	描述
create(Context context, int resId)	根据资源创建 MediaPlayer 对象
create(Context context, Uri uri)	根据URI创建MediaPlayer对象
new MediaPlayer()	创建MediaPlayer对象
setDataSource(String path)	根据文件路径设置MediaPlayer对象的数据源
setDataSource(Context context, Uri uri)	根据URI设置MediaPlayer 对象的音频源
prepare()	让 MediaPlayer 对象做好准备（同步）
prepareAsync()	让MediaPlayer对象做好准备（异步）
reset()	重置MediaPlayer对象
release()	释放 MediaPlayer 对象所占用的资源

音频播放：MediaPlayer

- 创建MediaPlayer对象（补充）
  - create()方法本质上会依次调用new MediaPlayer()、setDataSource()和prepare()。
  - 调用setDataSource()之前， MediaPlayer 对象需要处于空闲状态，即调用new MediaPlayer()或reset()之后的状态。
  - 调用setDataSource()之后，需要先调用prepare()或prepareAsync()，之后才能再调用start()。

音频播放：MediaPlayer

- 控制播放

方法	描述
start()	开始或继续播放
pause()	暂停播放
stop()	停止播放
seekTo(int msec)	跳转到指定位置播放

音频播放： MediaPlayer

- 获取播放状态

方法	描述
isPlaying()	判断 是否正在播放
getCurrentPosition()	获取当前播放的位置
getDuration()	获取音频文件的总时长

学习内容

12.1 图像与绘图

12.2 动画

12.3 音频播放

12.4 视频播放

主要内容

视频播放

- Android中播放视频有两种方式：
  - 使用VideoView控件播放视频。
  - 使用MediaPlayer + SurfaceView控件播放视频。其中，MediaPlayer 用于控制播放视频的播放，SurfaceView 用于显示视频内容。

视频播放：VideoView

- VideoView是一个可以播放视频的控件，通过调用setVideoURI()或setVideoPath()来加载视频文件，然后调用start()来播放视频。
- VideoView通常和MediaController搭配使用。
- MediaController是一个标准的控制器，为VideoView提供了一个控制面板，包括播放、暂停、快进、后退按钮和进度条等。
  - 通常MediaController显示在VideoView的底部。

视频播放：VideoView

- VideoView类的一些主要方法：

方法	描述
setVideoURI(Uri uri)	根据URI加载视频文件
setVideoPath(String path)	根据文件路径加载视频文件
start()	开始或恢复播放
pause()	暂停播放
seekTo(int msec)	跳转到指定位置
getCurrentPosition()	获取当前播放进度
getDuration()	获取视频的总时长
isPlaying()	检查视频是否正在播放
setMediaController(MediaController)	设置 MediaController，为 VideoView 提供播放控制面板

视频播放：VideoView

- MediaController的一些主要方法：

方法	描述
new MediaController(Context context)	构造一个新的 MediaController 实例
setAnchorView(View view)	设置 MediaController 的锚定视图，用于确定其显示位置
show()	显示 MediaController
hide()	隐藏 MediaController

视频播放：MediaPlayer+SurfaceView

- 通过MediaPlayer+SurfaceView可以播放视频，并实现开始播放、暂停播放、停止播放等功能。
- MediaPlayer 用于控制播放视频的播放，SurfaceView 用于显示视频内容。
- 使用SurfaceView时，需要为SurfaceView的SurfaceHolder添加Callback监听器。SurfaceHolder是一个接口，提供了访问和控制SurfaceView的方法。
- SurfaceHolder包括三个回调方法：
  - surfaceCreated()：当Surface被创建时回调
  - surfaceChanged()：当Surface大小或格式发生改变时调用
  - surfaceDestroyed()：当Surface被销毁时回调

课程总结

12.1 图像与绘图

12.2 动画

12.3 音频播放

12.4 视频播放