Note: Reading this document is not required to understand the course content. Rather, it helps give you a deeper understanding of the topic covered.

## Vector

Drawables that allow you to create sharp, flexible and density-independent images by defining a series of drawing commands. It uses a concept similar to SVG on the web.

## VectorDrawable

VectorDrawable, implementation of vector graphics in Android, was introduced in Android Lollipop and it allows you to create density-independent images by defining a series of drawing commands. It has concept similar to SVG on the web.

Using vectors has several advantages:

- Image scalability
- No need for creating multiple versions of images (as in bitmaps)
- Less space taken in the resources and on the device
- Simplifies maintenance

Vector definition:

- Has intrinsic size in dp like 36dp

- Has view port size which is used by the path

- Has a path which defines the drawing

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:height="64dp"
   android:width="64dp"
```

```
    android:viewportHeight="600"
    android:viewportWidth="600" >
       <path
          android:name="mypath"
          android:fillColor="#000000"
          android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
 </vector>
```

- Can have clip-path:

```
<vector>
 <group android:name=" group1" >
  <clip-path
   android:name=" clip"
   android:pathData=" ..." />

 </group>
</vector>
```

Clip path is defined for a group to draw a subsection of a group or vector. A clip-path cannot be defined for an individual path.

- Can have trim path:

Draw only part of a path by specifying a percentage e.g. 0.1, 0.5, and so on.

```
<path
android:pathData=" ..."
android:trimPathStart=" 0.1"
android:trimPathEnd=" 0.8"
/>
```

- Theme colors:

Theme colors can be used in two ways:

1- Applying a theme color as a tint (Tinted to right color at run-time):

```
<vector
android:tint=" ?attr/colorButtonNormal"
...
/>
```

2- Using theme colors as fillColor or strokeColor:

```
<vector
android:fillColor=" ?attr/colorPrimary"
```

- ColorStateList:

You can have different color for your vector paths based on selectors. See below:

```
<!-- state color defined: res/color/btn_selector -->
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_focused="true" android:color="#FFF03AED"/>
   <item android:color="?attr/colorButtonNormal"/>
</selector>

<!-- Use selector for drawable -->
<vector
  android:fillColor="color/btn_selector"
…
/>
```

Now set this drawable as your view's background. Make sure the view has clickable and focusable attributes set to true.

- Gradients:

Gradients support three types of gradients: Linear, radial, and sweep.

Gradients can be defined as color resources and then used to color vector paths.

```
<gradient  xmlns:android="http://schemas.android.com/apk/res/android"
android:startColor="@color/colorPrimary"
       android:endColor="@color/colorAccent"
       android:type="radial"
       android:gradientRadius="0.5"/>
```

## Vector Images in Android

Lollipop has two classes to support vector graphics:

- VectorDrawable: Defines a static drawable object
- AnimatedVectorDrawable: Can add animation to the properties of a vector drawable
- Vectors are defined as XML (tree structure as in SVG format):
    - Geometry is defined in path tags.

## How to Create a Vector Drawable

Drawing tools that export SVG images:

- Import SVG via Vector Asset Studio (Android Studio tool): via File→New→Vector Asset
- Inkscape
- LibreOffice Draw
- etc.

## Considerations

- Limit vector images to 200dp×200dp
- The initial loading of a vector image can cost more CPU usage than corresponding raster image.
- VectorDrawable is suitable for detailed images.
- VectorDrawable is best for iconography and simple drawings.

## AnimatedVectorDrawable

- A class in Android that allows animating vector drawables
- AnimatedVectorDrawable allows animating any property of a path (in vector) or group of paths

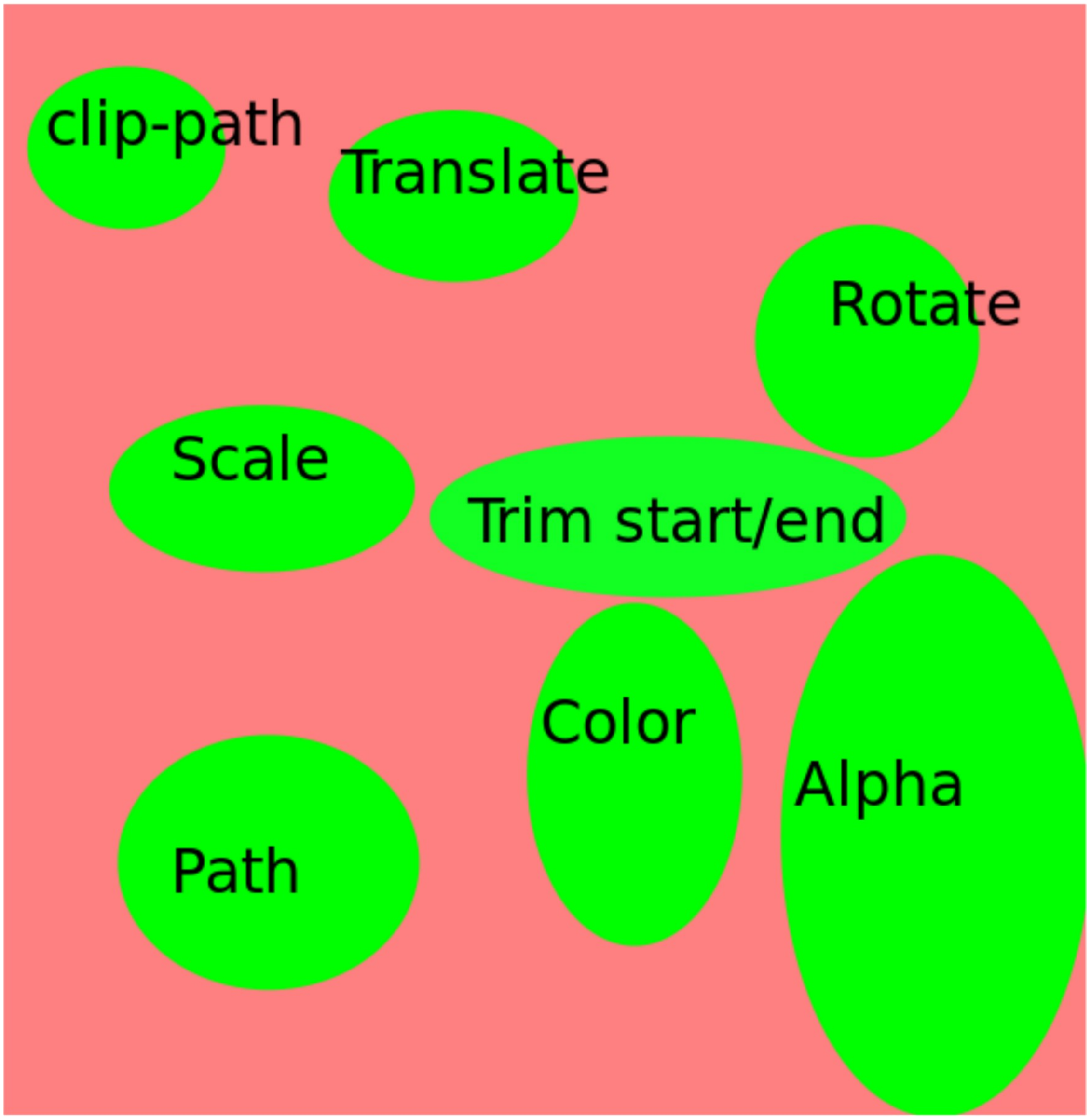


Figure 1: Many types of animations supported by AVDs

- AnimatedVectorDrawable uses names to find a target object inside a VectorDrawable.
- The target object to be animated can be either a group, a tag, or a root tag (with unique names)

• AnimatedVectorDrawable: Combines VectorDrawable and ObjectAnimator

ObjectAnimator can be used to animate pathData.

• Starting from API 25, AnimatedVectorDrawable runs on RenderThread, which means smoother running of AnimatedVectorDrawable animations.

Path Morphing Requirements:

The two paths must use the same number of types of drawing commands.

## How to Create AnimatedVectorDrawable

AnimatedVectorDrawable can be defined in either three separate XML files or one XML.

1- An XML defining vector to be animated

2- XML files defining animators that will be used by the AVD

2- XML defining vector attribute to be animated, with which animators and on which target.

A Simple Example follows.

1- Vector:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
  <group
      android:name="rotationGroup"
      android:pivotX="300.0"
      android:pivotY="300.0"
      android:rotation="45.0" >
    <path
        android:name="v"
        android:fillColor="#000000"
        android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
  </group>
</vector>
```

2- Animators:

```
<!-- rotation.xml -->
<objectAnimator
    android:duration="@android:integer/config_longAnimTime"
    android:valueFrom="0"
        android:valueTo="360"
        android:propertyName="rotation"
        android:repeatCount="infinite"
        android:valueType="floatType" xmlns:android="http://schemas.android.com/apk/res/android"/>

<!-- path_morph.xml -->
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <objectAnimator
      android:duration="3000"
      android:propertyName="pathData"
      android:valueFrom="M300,70 l 0,-70 70,70 0,0 -70,70z"
      android:valueTo="M300,70 l 0,-70 70,0  0,140 -70,0 z"
      android:valueType="pathType"
      android:repeatCount="infinite"
  />


</set>
```

3- Animated vector drawable:

```
<?xml version="1.0" encoding="utf-8"?>
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
         android:drawable="@drawable/my_vector" >
  <target
      android:name="rotationGroup"
      android:animation="@animator/rotation" />
  <target
      android:name="v"
      android:animation="@animator/path_morph" />
</animated-vector>
```

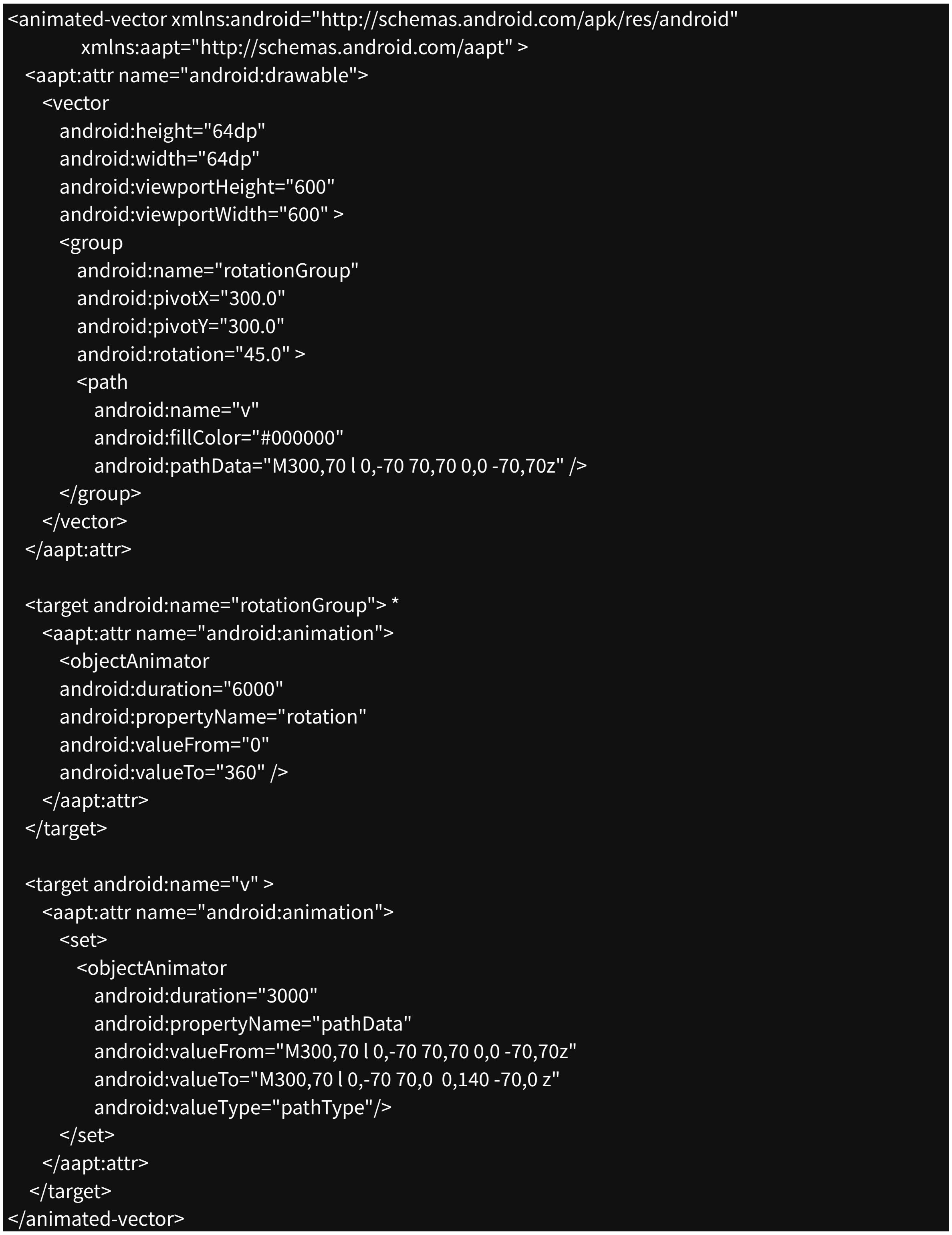
How to animate an animated vector drawable in code:

The following shows how to start animation on an animated vector drawable that has been used as a view’s background.

```
AnimatedVectorDrawable avd = (AnimatedVectorDrawable) imageView.getBackgroundDrawable();
avd.start();
```

## Define the AVD in A Single File

Using the AAPT (Android Asset Packing Tool) tool, we can merge vector and animtor in a single file.



```
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:aapt="http://schemas.android.com/aapt" >
  <aapt:attr name="android:drawable">
    <vector
      android:height="64dp"
      android:width="64dp"
      android:viewportHeight="600"
      android:viewportWidth="600" >
      <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
          android:name="v"
          android:fillColor="#000000"
          android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
      </group>
    </vector>
  </aapt:attr>

  <target android:name="rotationGroup"> *
    <aapt:attr name="android:animation">
      <objectAnimator
      android:duration="6000"
      android:propertyName="rotation"
      android:valueFrom="0"
      android:valueTo="360" />
    </aapt:attr>
  </target>

  <target android:name="v" >
    <aapt:attr name="android:animation">
      <set>
        <objectAnimator
          android:duration="3000"
          android:propertyName="pathData"
          android:valueFrom="M300,70 l 0,-70 70,70 0,0 -70,70z"
          android:valueTo="M300,70 l 0,-70 70,0  0,140 -70,0 z"
          android:valueType="pathType"/>
      </set>
    </aapt:attr>
   </target>
</animated-vector>
```

## How to Use AnimatedVectorDrawable in Android Applications

There are two ways you can use AnimatedVectorDrawable in Android apps. Using the AVD as drawable source of an ImageView or as background of a view.

## Use as ImageView Source

### XML declaration

```
<ImageView
android:id=” @+id/myImage”
android:layout_width=” ...”
android:layout_height=” ...”
android:src=” @drawable/my_avd”
/>
```

### Java Code

```
ImageView  image = findViewById(R.id.myImage)
AnimatedVectorDrawable avd = (AnimatedVectorDrawable)image.getDrawable();
avd.start(); //Start the avd animation
```

### Kotlin code

```
val image = findViewById<ImageView>(R.id.myImage)
val avd = image.drawable as AnimatedVectorDrawable
avd.start() //Start the avd animation
```

## Use As Background

### XML declaration

```
<ImageView
android:id=” @+id/myImage”
android:layout_width=” ...”
android:layout_height=” ...”
android:background=” @drawable/my_avd”
/>
```

### Java Code

```
ImageView image = findViewById(R.id.myImage)
AnimatedVectorDrawable avd = (AnimatedVectorDrawable)image.getBackground();
avd.start();//Start the avd
```

### Kotlin Code

```
val image = findViewById<ImageView>(R.id.myImage)
val avd = image.background as AnimatedVectorDrawable
avd.start() //Start avd
```