Drawing and Working with Animation

Introduction

Android Drawing App are support to following things:

- Draw – Users will be able to draw on a blank canvas (whiteboard).
- Erase – Users will be able to erase what has been drawn.
- Undo – Users will be able to undo and redo drawing paths.
- Color – Users will be able to draw using a color of their choice from at least these colors: black, dark gray, light gray, blue, red, and green, orange, yellow.
- Share – Users will be able to capture a screen shot and email it to a friend.

Paint applications are become famous thanks to Microsoft Paint, well known as simply Paint or Paintbrush. It was a simple computer graphics application included with all versions of Microsoft Windows. In this chapter, you are going to discover how to create a Paint Application for Android which will let users to draw on the screen with their fingers.

- Draw paths with fingers on the screen
- Normal mode
- Emboss mode
- Blur mode
- Clear option to remove all paths on the screen

Finger Path Object: The first step is to create a FingerPath Object to represent a path drawn with the finger on the screen. Our FingerPath class will have several fields letting us to define:

- Colour of the path
- Emboss mode or no
- Blur mode or no
- Stroke width of the path
- Path object from the standard SDK representing the path drawn

In this module we learnt drawing, 2d animation

Animation and Type

Android Animation is used to give the UI a rich look and feel. Animations in android apps can be performed through XML or android code. we'll go with XML codes for adding animations into our application.

Android Animation

Animation in android apps is the process of creating motion and shape change. The basic ways of animation that we'll look upon in this modules are:

- Fade In Animation
- Fade Out Animation
- Cross Fading Animation
- Blink Animation
- Zoom In Animation
- Zoom Out Animation
- Rotate Animation
- Move Animation
- Slide Up Animation
- Slide Down Animation
- Bounce Animation
- Sequential Animation
- Together Animation

Android Animation Example XML

We create a resource directory under the res folder names anim to keep all the xml files containing the animation logic. Following is a sample xml file showing an android animation code logic.

sample_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<scale xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_decelerate_interpolator"
  android:duration="300"
  android:fillAfter="true"
  android:fromXScale="0.0"
  android:fromYScale="0.0"
  android:toXScale="1.0"
  android:toYScale="1.0" />
```

android:interpolator : It is the rate of change in animation. We can define our own interpolators using the time as the constraint. In the above xml code an inbuilt interpolator is assigned

android:duration : Duration of the animation in which the animation should complete. It is 300ms here. This is generally the ideal duration to show the transition on the screen.

The start and end of the animation are set using:

```
android:fromTRANSFORMATION
android:toTRANSFORMATION
```

transformation : is the transformation that we want to specify. In our case we start with an x and y scale of 0 and end with an x and y scale of 1

android:fillAfter : property specifies whether the view should be visible or hidden at the end of the animation. We've set it visible in the above code. If it sets to false, the element changes to its previous state after the animation

android:startOffset : It is the waiting time before an animation starts. This property is mainly used to perform multiple animations in a sequential manner

android:repeatMode : This is useful when you want the animation to be repeat

android:repeatCount : This defines number of repetitions on animation. If we set this value to infinite then animation will repeat infinite times

Animation Examples

Let's create android animation application, open android studio and Execute project creation wizard with own credentials,

Loading Animation when UI widget is clicked: Our aim is to show an animation when any widget(lets say TextView) is clicked. For that we need to use the Animation Class. The xml file that contains the animation logic is loaded using AnimationUtils class by calling the loadAnimation() function. The below snippet shows this implementation.

```
Animation animation;
animation = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.sample_animation);
```

To start the animation we need to call the startAnimation() function on the UI element as shown in following : sampleTextView.startAnimation(animation);

Here we perform the animation on a textview component by passing the type of Animation as the parameter.

Setting the Animation Listeners

This is only needed if we wish to listen to events like start, end or repeat. For this the activity must implement AnimationListener and the following methods need to overridden.

onAnimationStart : This will be triggered once the animation started
onAnimationEnd : This will be triggered once the animation is over

onAnimationRepeat : This will be triggered if the animation repeats