

Experiment: 3.3

Student Name: Yash Kumar

UID: 20BCS9256

Branch: BE-CSE

Section/Group: 616 'B'

Semester: 5th

Subject Name: WMS Lab

Subject Code: 21CSP-338

Aim: Create animations and graphical primitives in Android environment

Objective: To draw 2D graphics and Animation in android application.

Software/Hardware Requirements: Android Studio

Discussion:

Introduction: Android graphics provides low level graphics tools such as canvases, color, filters, points and rectangles which handle drawing to the screen directly.

- Android provides a huge set of 2D-drawing APIs that allow you to create graphics.
- Android has got visually appealing graphics and mind-blowing animations.
- The Android framework provides a rich set of powerful APIs for applying animation to UI elements and graphics as well as drawing custom 2D and 3D graphics.

Following are the three animation systems used in Android applications:

1. Property Animation
2. View Animation
3. Drawable Animation

Property Animation

- Property animation is the preferred method of animation in Android.
- This animation is the robust framework which lets you animate any properties of any objects, view or non-view objects.
- The **android.animation** provides classes which handle property animation.

2. View Animation

- View Animation is also called as **Tween Animation**.
- The **android.view.animation** provides classes which handle view animation.
- This animation can be used to animate the content of a view.
- It is limited to simple transformation such as moving, re-sizing and rotation, but not its background color.

3. Drawable Animation

- Drawable animation is implemented using the AnimationDrawable class.
- This animation works by displaying a running sequence of 'Drawable' resources that is images, frame by frame inside a view object.

Reading Material (add reference links along with material):

Android Simple Graphics Example

The android.graphics.Canvas can be used to draw graphics in android. It provides methods to draw oval, rectangle, picture, text, line etc.

The android.graphics.Paint class is used with canvas to draw objects. It holds the information of color and style.

Canvas

- Android graphics provides low level graphics tools such as canvases, color, filters, points and rectangles which handle drawing to the screen directly.
- The Android framework provides a set of 2D-DRAWING APIs which allows user to provide own custom graphics onto a canvas or to modify existing views to customize their look and feel.

There are two ways to draw 2D graphics,

1. Draw your animation into a View object from your layout.
2. Draw your animation directly to a Canvas.

Some of the important methods of Canvas Class are as follows

- i. drawText()
- ii. drawRoundRect()
- iii. drawCircle()
- iv. drawRect()
- v. drawBitmap()
- vi. drawARGB()

- You can use these methods in onDraw() method to create your own custom user interface.
- Drawing an animation with a View is the best option to draw simple graphics that do not need to change dynamically and are not a part of a performance-intensive game. It is used when user wants to display a static graphic or predefined animation.
- Drawing an animation with a Canvas is better option when your application needs to re-draw itself regularly.

Steps/Method/Coding:

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

```
<RelativeLayout
```

```
xmlns:and
```

```
roid="http://schemas.android.com/apk/res/android"
```

```
xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
```

```
tools:context=".MainActivity">
```

```
<ImageView
```

```
android:id="@+id/imageview"
```

```
android:layout_width="200dp"
```

```
android:layout_height="200dp"
```

```
android:layout_centerHorizontal="true"

" android:layout_marginTop="40dp"

android:contentDescription="@string/app_name"
android:src="@drawable/gfgimage" />
```

```
<LinearLayout
```

```
    android:id="@+id/linear1"

    android:layout_width="match_parent"
    "

    android:layout_height="wrap_content"
    "

    android:layout_below="@id/
    imageview"

    android:layout_marginTop="30dp"

    android:orientation="horizontal"

    android:weightSum="3">
```

```
<!--To start the blink animation of the image-->
```

```
<Button
```

```
    android:id="@+id/BTNblink"

    style="@style/TextAppearance.AppCompat.Widget.Button"

    android:layout_width="0dp"

    android:layout_height="wrap_content"

    android:layout_margin="10dp"

    android:layout_weight="1"

    android:padding="3dp"
```

```
android:text="@string/blink"
```

```
android:textColor="@color/white" />
```

```
<!--To start the rotate animation of the image-->
```

```
<Button
```

```
    android:id="@+id/BTNrotate"
```

```
    style="@style/TextAppearance.AppCompat.Widget.Button"
```

```
    android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_margin="10dp"
```

```
    android:layout_weight="1"
```

```
    android:padding="3dp"
```

```
    android:text="@string/clockwise"
```

```
    android:textColor="@color/white" />
```

```
<!--To start the fading animation of the image-->
```

```
<Button
```

```
    android:id="@+id/BTNfade"
```

```
    style="@style/TextAppearance.AppCompat.Widget.Button"
```

```
    android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_margin="10dp"
```

```
    android:layout_weight="1"
```

```
    android:padding="3dp"
```

```
    android:text="@string/fade"
```

```
    android:textColor="@color/white" />
```

</LinearLayout>

<LinearLayout

android:id="@+id/linear2"

android:layout_width="match_parent"

"

android:layout_height="wrap_content"

"

android:layout_below="@id/linear1"

android:layout_marginTop="30dp"

android:orientation="horizontal"

android:weightSum="3">

<!--To start the move animation of the image-->

<Button

android:id="@+id/BTNmove"

style="@style/TextAppearance.AppCompat.Widget.Button"

android:layout_width="0dp"

android:layout_height="wrap_content"

android:layout_margin="10dp"

android:layout_weight="1"

android:padding="3dp"

android:text="@string/move"

android:textColor="@color/white" />

<!--To start the slide animation of the image-->

<Button

```
    android:id="@+id/BTNslide"
    style="@style/TextAppearance.AppCompat.Widget.Button"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:layout_weight="1"
    android:padding="3dp"
    android:text="@string/slide"
    android:textColor="@color/white" />
```

<!--To start the zoom animation of the image-->

<Button

```
    android:id="@+id/BTNzoom"
    style="@style/TextAppearance.AppCompat.Widget.Button"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:layout_weight="1"
    android:padding="3dp"
    android:text="@string/zoom"
    android:textColor="@color/white" />
```

</LinearLayout>

<!--To stop the animation of the image-->

<Button

android:id="@+id/BTNstop"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_below="@id/linear2"

android:layout_marginLeft="30dp"

android:layout_marginTop="30dp"

android:layout_marginRight="30dp"

android:text="@string/stop_animation" />

</RelativeLayout>

Output:

