

**Name: Aniket Kumar**

**UID:20BCS5306**

**Branch:BE-CSE**

**Section/Group:20BCS\_WM-703B**

**Semester: 5**

**Subject :Web and Mobile Security**

### **Experiment - 10**

**Aim:**Create animations and graphical primitives in Android environment

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

**Software/Hardware Requirements:**Android Studio

**Tools to be used:**Android Studio

**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 API for applying animation to UI elements and graphics as well as drawing custom 2D and 3Dgraphics. 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,viewor 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 •  
Thisanimationcanbeusedt oanimation.animatethecontentofa 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 are 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 be changed dynamically and are not a part of a performance-critical intensive game. It is used when the user wants to display a static graphic or a predefined animation.
- Drawing an animation with a Canvas is a better option when your application needs to redraw itself regularly.

### Steps/Method/Coding:

- Open Eclipse or Android Studio and select a new Android project
- Give project name and select next
- Choose the Android version. Choose the lowest Android version (Android 2.2) and select next
- Enter the package name. Package name must be two words separated by a comma and click finish
- Go to package explorer in the left-hand side. Select our project.
- Go to res folder and select layout. Double-click the main.xml file.  
Don't change anything in layout. Leave as default.

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

```
<RelativeLayout xmlns:android="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"/>
```

```
<LinearLayoutandroid:id="@+id/linear1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_below="@id/imageview"
```

**android:orientation="horizontal"**

**android:weightSum="3">**

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

**<Button android:id="@+id/BTN**

**blink"**

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

**android:layout\_width="0dp" android:layout\_height="wrap**

**\_content" android:layout\_margin="10dp"**

**android:layout\_weight="1" android:pa**

**dding="3dp" android:text="@string/b**

**link" android:textColor="@color/whit**

**e"/>**

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

**<Button android:id="@+id/BTNr**

**otate"**

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

**android:layout\_width="0dp" android:layout\_height="wrap**

**\_content" android:layout\_margin="10dp"**

**android:layout\_weight="1" android:pa**

```
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-->
```

```

<Buttonandroid:id="@+id/BTN move"
    style="@style/TextAppearance.AppCompat.Widget.Button"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"android
    :layout_weight="1"android:padding="
    3dp"android:text="@string/move"and
    roid:textColor="@color/white" />

```

```

<!--Tostarttheslideanimationof theimage-->

```

```

<Buttonandroid:id="@+id/BTN
    slide"
    style="@style/TextAppearance.AppCompat.Widget.Button"
    android:layout_width="0dp"android:layout_height="wrap
    _content"android:layout_margin="10dp"
    android:layout_weight="1"android:pa
    dding="3dp"android:text="@string/sl
    ide"android:textColor="@color/white
    "/>

```

```

<!--Tostart thezoomanimationoftheimage-->

```

```

<Buttonandroid:id="@+id/BTN zoom"
    style="@style/TextAppearance.AppCompat.Widget.Button"
    android:layout_width="0dp"android:layout_height="wrap_

```

```

        content"android:layout_margin="10dp"
        android:padding="3dp"android:text=

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

</LinearLayout>

```

```

<!--Tostopthe animationof theimage-->

<Buttonandroid:id="@+id/BT Nstop"
        android:layout_width="match_parent"an
        droid:layout_height="wrap_content"andr
        oid:layout_below="@id/linear2"android:l
        ayout_marginLeft="30dp"android:layout
        _
        marginTop="30dp"android:layout_margi
        nRight="30dp"android:text="@string/sto
        p_animation"/>

</RelativeLayout>

```

**Learning Outcomes:** Learned Basics of Android, Android Layouts and Widgets and Communication and Media.