Name of Student : Neha Yadav		
Roll Number : 62		**LAB Assignment Number :** 06
Title of LAB Assignment : To perform the animation on an image and to apply various filters on an image. A) Perform the following animation on the image: 1. Move. 2. Rotate. 3. Expand. B) Apply the following effects on the image: 1. Brightness. 2. Darkness. 3. Grayscale		
DOP : 13th September 2024		**DOS :** 27th September 2024
CO Mapped : CO2, CO4	**PO Mapped :** PO2, PO3, PO5, PSO1, PSO2	**Signature :**

AIM:

To perform the animation on an image and to apply various filters on an image.
A) Perform the following animation on the image:
1. Move.
2. Rotate.
3. Expand.

B) Apply the following effects on the image:
1. Brightness.
2. Darkness.
3. Grayscale

THEORY:

Image Animation in Android

1. Move: Image movement, or translation, is a common animation effect where an image shifts from one position to another. In Android, this is achieved through the `TranslateAnimation` class. This animation can be used to move the image along the X-axis, Y-axis, or both. The key parameters include `fromXDelta`, `toXDelta`, `fromYDelta`, and `toYDelta`, which define the starting and ending points of the movement. This effect is useful in creating dynamic interfaces where elements need to appear to slide or drift into position.

2. Rotate: Rotation involves changing the angle of an image around a specific point, usually its center. The `RotateAnimation` class in Android is used for this purpose. It allows the image to spin or rotate from a starting angle to an ending angle. You can specify the pivot point using `pivotX` and `pivotY`, which are often set to the center of the image to ensure smooth rotation. This effect is commonly used for creating interactive elements like rotating icons or buttons.

3. Expand: Expanding an image involves scaling it up or down, altering its size without changing its aspect ratio. This is managed using the `ScaleAnimation` class. It scales the image's width and height from an initial value to a final value. Key parameters include `fromXScale`, `toXScale`, `fromYScale`, and `toYScale`, which define the scaling factors. This animation is useful for emphasizing or de-emphasizing visual elements, such as enlarging a photo when clicked.

Image Filters in Android

1. Brightness: Adjusting brightness involves modifying the lightness or darkness of the image. This is achieved by altering the color values of the image pixels. In Android, a `ColorMatrix` is used to adjust brightness, which involves setting the matrix to scale the RGB values of each pixel. Increasing brightness involves scaling up these values, while decreasing involves scaling them down. This adjustment is essential for improving visibility or achieving a particular mood in an image.

2. Darkness: Applying darkness is essentially the inverse of adjusting brightness. By reducing the color values of the image, it appears darker. This can be achieved by using a color matrix with a scaling factor less than one. This effect can be used to create a somber or dramatic look in images or to reduce the contrast in bright areas.

3. Grayscale: Converting an image to grayscale removes all color information and replaces it with varying shades of gray. This is done by setting the saturation of the image to zero using a `ColorMatrix`. The grayscale filter averages the color values to calculate the luminance, resulting in shades of gray that represent the intensity of light. This effect is useful for creating a classic or artistic look and for simplifying images when color information is not necessary.

CODE:

A)

MainActivity.java

```
package com.example.imageanimationapp;

import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationSet;
import android.view.animation.RotateAnimation;
import android.view.animation.ScaleAnimation;
import android.view.animation.TranslateAnimation;
import android.widget.Button;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
```

```
private ImageView imageView;
private Button startButton;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    imageView = findViewById(R.id.imageView);
    startButton = findViewById(R.id.startButton);

    startButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startAnimations();
        }
    });
}

private void startAnimations() {
    // Create and configure move animation
    TranslateAnimation moveAnimation = new TranslateAnimation(
            0, 200, // Move from X=0 to X=200
            0, 0    // No vertical movement
    );
    moveAnimation.setDuration(3000); // 3 seconds for slower animation
    moveAnimation.setFillAfter(true); // Retain position after animation

    // Create and configure rotate animation
    RotateAnimation rotateAnimation = new RotateAnimation(
            0, 360, // Rotate from 0 to 360 degrees
            Animation.RELATIVE_TO_SELF, 0.5f, // Pivot X (center of the view)
            Animation.RELATIVE_TO_SELF, 0.5f  // Pivot Y (center of the view)
    );
    rotateAnimation.setDuration(3000); // 3 seconds for slower animation
    rotateAnimation.setFillAfter(true); // Retain rotation after animation

    // Create and configure expand animation
    ScaleAnimation scaleAnimation = new ScaleAnimation(
            1, 1.5f, // Scale from 1x to 1.5x in X direction
```

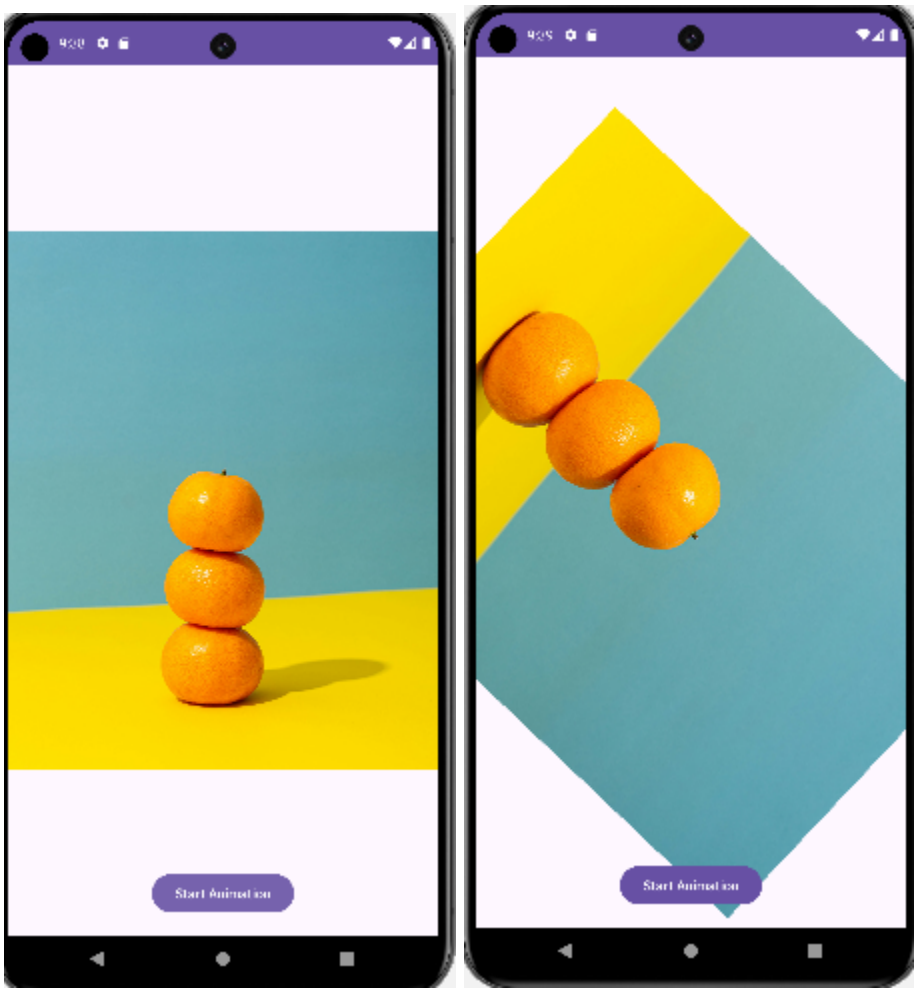
```
            1, 1.5f, // Scale from 1x to 1.5x in Y direction
            Animation.RELATIVE_TO_SELF, 0.5f, // Pivot X (center of the view)
            Animation.RELATIVE_TO_SELF, 0.5f  // Pivot Y (center of the view)
        );
        scaleAnimation.setDuration(3000); // 3 seconds for slower animation
        scaleAnimation.setFillAfter(true); // Retain size after animation

        // Combine all animations into an AnimationSet
        AnimationSet animationSet = new AnimationSet(true);
        animationSet.addAnimation(moveAnimation);
        animationSet.addAnimation(rotateAnimation);
        animationSet.addAnimation(scaleAnimation);

        // Start animations
        imageView.startAnimation(animationSet);
    }
}
```

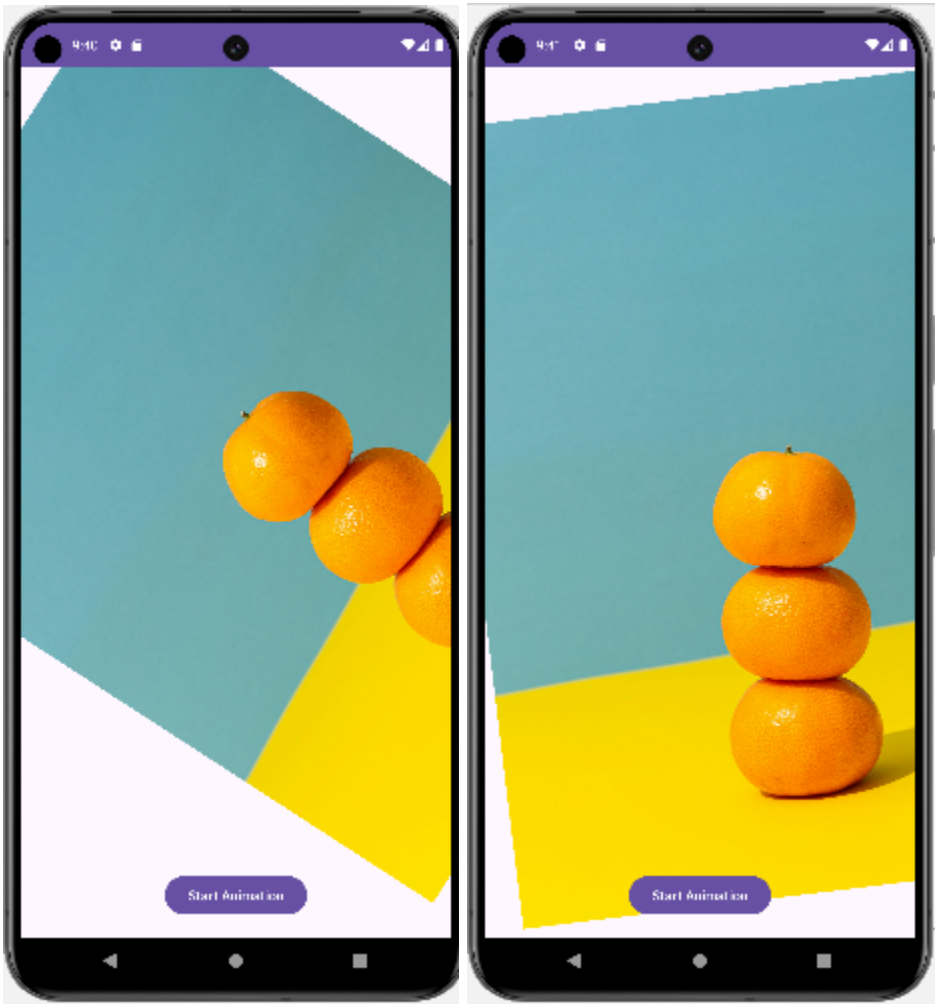
Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/image"
        android:layout_centerInParent="true" />

    <Button
        android:id="@+id/startButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Animation"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp"/>
```

</RelativeLayout>

OUTPUT :




Start Animation
Start Animation

B)

MainActivity.java

```
package com.example.imageanimationapp;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.ColorMatrix;
import android.graphics.ColorMatrixColorFilter;
import android.graphics.Paint;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

  private ImageView imageView;
  private Button brightnessButton;
  private Button darknessButton;
  private Button grayscaleButton;
  private Bitmap originalBitmap;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    imageView = findViewById(R.id.imageView);
    brightnessButton = findViewById(R.id.brightnessButton);
    darknessButton = findViewById(R.id.darknessButton);
    grayscaleButton = findViewById(R.id.grayscaleButton);

    // Load and scale the original bitmap
    originalBitmap = decodeSampledBitmapFromResource(R.drawable.image, 800,
800); // Adjust size as needed
    imageView.setImageBitmap(originalBitmap);
```

```
        // Set up button click listeners
        brightnessButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                imageView.setImageBitmap(applyBrightness(originalBitmap, 1.5f)); // Adjust
the factor as needed
            }
        });

        darknessButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                imageView.setImageBitmap(applyBrightness(originalBitmap, 0.5f)); // Adjust
the factor as needed
            }
        });

        grayscaleButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                imageView.setImageBitmap(applyGrayscale(originalBitmap));
            }
        });
    }

    private Bitmap decodeSampledBitmapFromResource(int resId, int reqWidth, int
reqHeight) {
        final BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        BitmapFactory.decodeResource(getResources(), resId, options);

        options.inSampleSize = calculateInSampleSize(options, reqWidth, reqHeight);

        options.inJustDecodeBounds = false;
        return BitmapFactory.decodeResource(getResources(), resId, options);
    }

    private int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int
reqHeight) {
```

```
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;

        if (height > reqHeight || width > reqWidth) {
            final int halfHeight = height / 2;
            final int halfWidth = width / 2;

            while ((halfHeight / inSampleSize) >= reqHeight && (halfWidth / inSampleSize)
>= reqWidth) {
                inSampleSize *= 2;
            }
        }

        return inSampleSize;
    }

    private Bitmap applyBrightness(Bitmap original, float factor) {
        Bitmap result = Bitmap.createBitmap(original.getWidth(), original.getHeight(),
original.getConfig());
        ColorMatrix colorMatrix = new ColorMatrix();
        colorMatrix.set(new float[]{
                factor, 0, 0, 0, 0,
                0, factor, 0, 0, 0,
                0, 0, factor, 0, 0,
                0, 0, 0, 1, 0
        });

        Canvas canvas = new Canvas(result);
        Paint paint = new Paint();
        paint.setColorFilter(new ColorMatrixColorFilter(colorMatrix));
        canvas.drawBitmap(original, 0, 0, paint);
        return result;
    }

    private Bitmap applyGrayscale(Bitmap original) {
        Bitmap result = Bitmap.createBitmap(original.getWidth(), original.getHeight(),
original.getConfig());
        ColorMatrix colorMatrix = new ColorMatrix();
        colorMatrix.setSaturation(0); // Convert to grayscale
```

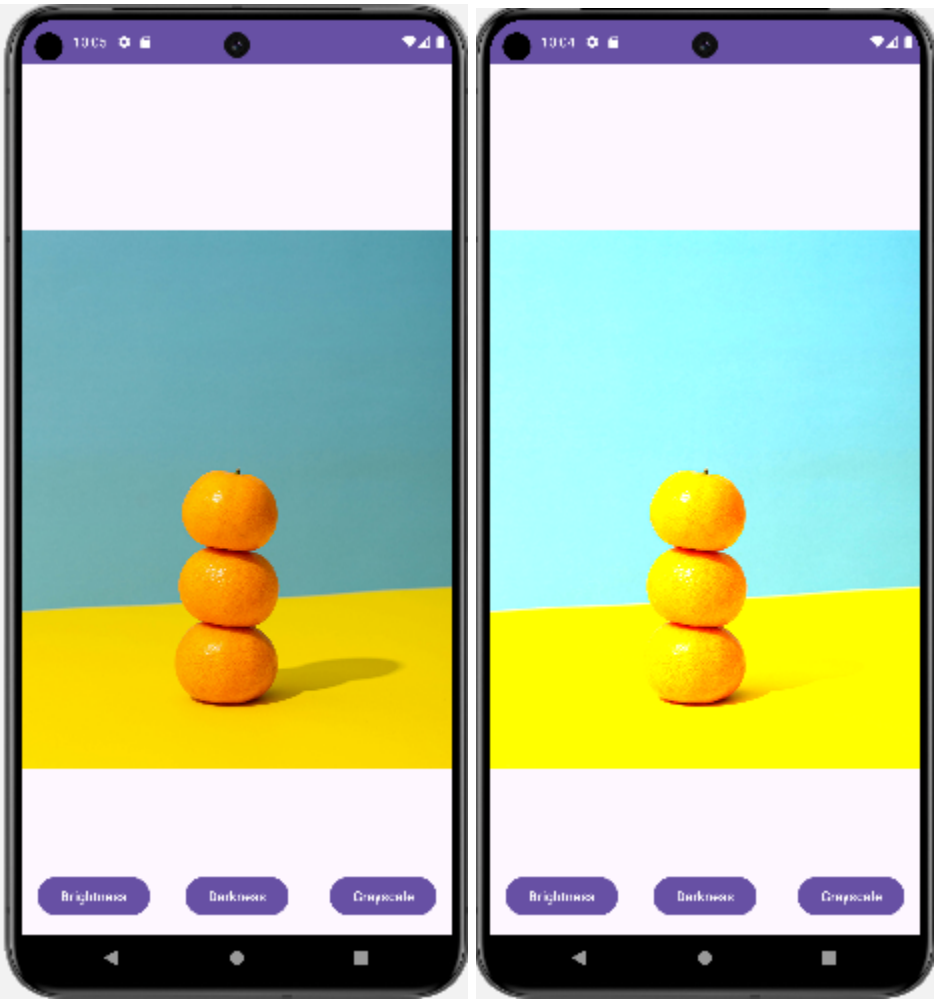
```
        Canvas canvas = new Canvas(result);
        Paint paint = new Paint();
        paint.setColorFilter(new ColorMatrixColorFilter(colorMatrix));
        canvas.drawBitmap(original, 0, 0, paint);
        return result;
    }
}
```

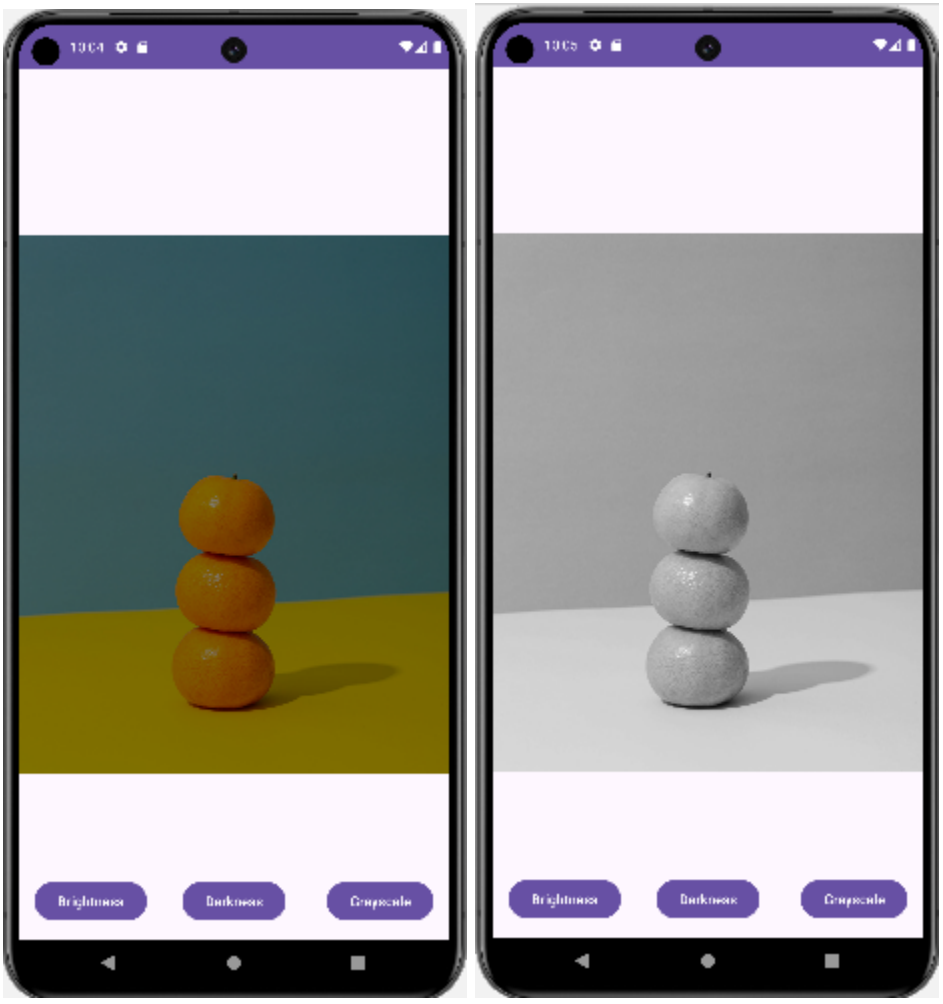
Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/image"
        android:layout_centerInParent="true" />

    <Button
        android:id="@+id/brightnessButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Brightness"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:layout_margin="16dp" />

    <Button
        android:id="@+id/darknessButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Darkness"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_margin="16dp" />
```

```
  <Button
     android:id="@+id/grayscaleButton"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Grayscale"
     android:layout_alignParentBottom="true"
     android:layout_alignParentEnd="true"
     android:layout_margin="16dp" />
</RelativeLayout>
```

OUTPUT:


Brightness
Darkness
Grayscale
Brightness
Darkness
Grayscale



CONCLUSION:

In this practical, we have explored the fundamental techniques of animating and applying filters to an image. By utilizing animations such as move, rotate, and expand, we can create dynamic visual effects. Meanwhile, applying filters like brightness, darkness, and grayscale allows us to enhance or modify the image's appearance, offering greater control over the visual output. These techniques are crucial in developing engaging and visually appealing Android applications.