Birzeit University
Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Advance Computer Systems Engineering Lab ENCS515

EXP. No. 5. Frame Animation and Tween Animation in Android

1. Objectives

- ❖ To provide a good understanding of how to use Frame animation and Tween animation in android on any View.
- ❖ To provide knowledge of some attributes of translate, rotate and scale tags.
- ❖ To provide familiarity with Animation Class.

2. Introduction

Animations add vivacity and personality to your apps. There are two types of animations that you can do with the Animation framework of Android: Frame animation and Tween animation.

2.1. Frame animation

Is series of frames is drawn one after the other at regular. creates an animation by showing a sequence of images in order with an AnimationDrawable.

2.2. Tween animation

Are simple transformations of position, size, rotation to the content of a View. Animation can be defined in XML that performs transitions such as rotating, fading, moving, and stretching on a graphic.

- Animation Class: Animation class is used to hold an animation which is loaded in it from the anim folder of resource directory. The animation reference variable holds the animation xml files loaded using AnimationUtils class.
- AnimationUtils: AnimationUtils class is used to fetch an animation file from anim folder by using loadAnimation method of this class which has two arguments: First argument defines context in which the animation is to be loaded which must be the context of given activity on which the animation is to be loaded. Second argument defines the id of the animation in resources file.
- Main attributes in Tween animation

- **xmlns:android:** This attribute must be defined in the root tag to get the schema path of this XML file.
- **android:pivotY and android:pivotY:** Used for specifying the center point of the rotation.
- **android:duration:** This attribute is used to define the duration in which the animation needs to be completed in milliseconds.
- **android:fromXDelta:** This attribute is used to define the starting point of translation animation on Xaxis.
- **android:toXDelta:** This attribute is used to define the ending point of translation animation on Xaxis. Values from -100 to 100 ending with "**%**", indicating a percentage relative to itself. Values from -100 to 100 ending in "**%p**", indicating a percentage relative to its parent. A float value with no suffix, indicating an absolute value.
- **android:fromXScale:** Starting X size offset, where 1.0 is no change.
- **android:toXScale:** Ending X size offset, where 1.0 is no change.
- **android:fromYScale:** Starting Y size offset, where 1.0 is no change.
- **android:toYScale:** Ending Y size offset, where 1.0 is no change.
- **android:pivotX:** The X coordinate to remain fixed when the object is scaled.
- **android:pivotY:** The Y coordinate to remain fixed when the object is scaled.

3. Procedure

You are going to build two different applications the first is by using frame animation and the second is by tween animation. You will need for the frame animation application several images. However, in tween one image is needed.

3.1. Frame animation Application

Create a new project with the name “Frame Animation Application”. This project will exchange between two images. The idea behind a frame animation is simple: You’ll be cycling through a series of images very quickly, just like an old movie. The “frame” refers to a single image. Thus, the first step in creating a custom frame animation is to create a sequence of images.

You have two options here: you can use XML drawable (such as shape drawable) or actual image files. For the sake of simplicity, you are going to use the following series of PNG images. You should make sure to have images sized appropriately for different screen densities.

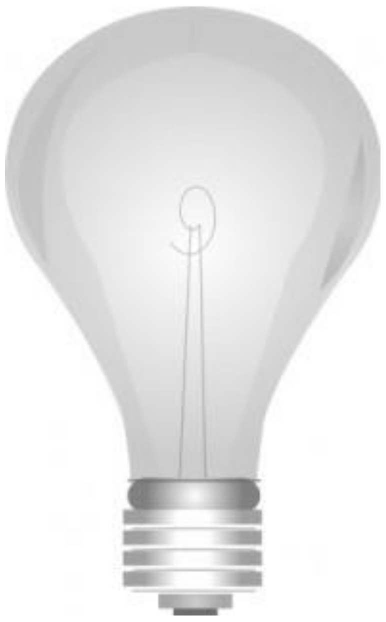


Figure 5.1 First Image

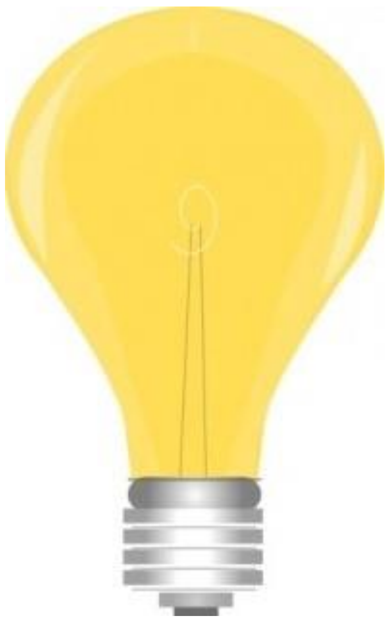


Figure 5.2 Second Image

- import those images into the res/drawable folder and call them off for Figure 5.1 and on for Figure 5.2 as shown in Figure 5.3
- Define an XML Drawable for our animation. We can use transition to make the Frame Animation. It just cycles through a sequence of provided images. Call this file “animation” and locate it in res/drawable as shown in Figure 5.3.
- The code below is to make transition between two images (on and off) this code should be in the animation.xml folder.

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/off" />
    <item android:drawable="@drawable/on" />
</transition>
```

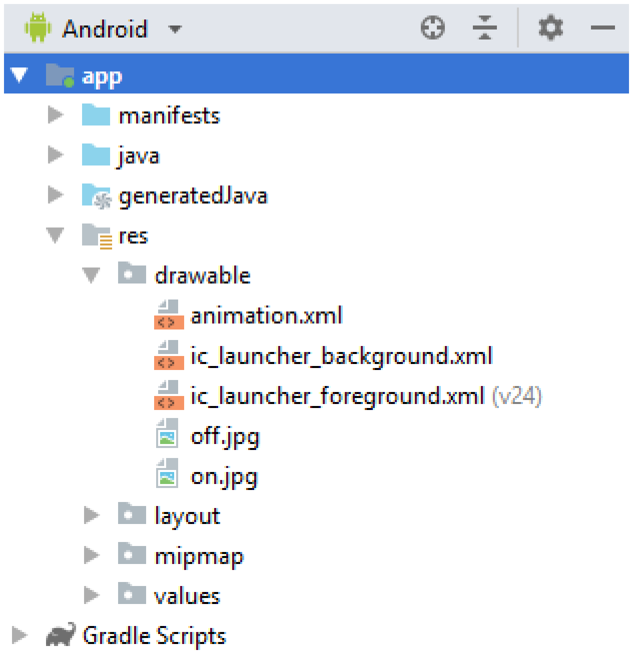


Figure 5.3 Animation xml Code File and On and Off Images Location Screen

- Add a button and an image view in the activity_main layout as shown in Figure 5.4.
- Add @drawable/animation to the source compact when adding the image view as shown in Figure 5.5.

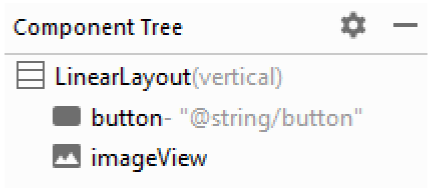


Figure 5.4 activity_main layout

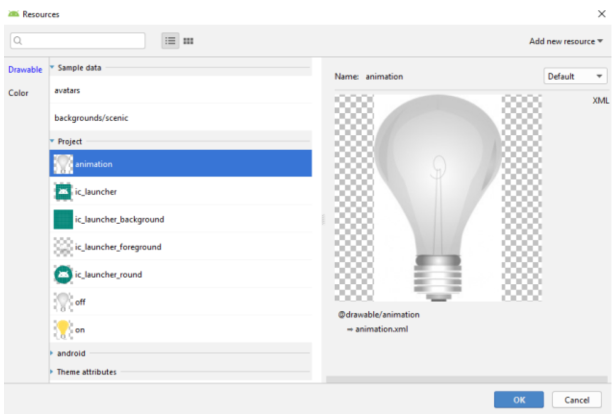


Figure 5.5 Adding the animation to the ImageView

- In the main activity java folder, you will make objects of Button and the ImageView and connect them with the button and the imageView in the layout folder as shown in the code below.
- Make a setOnClickListener method to the button.
- Make a TransitionDrawable object in the button click listener and start the transition by calling the startTransition method and pass the duration to this method in milli Second as shown in the code below.Figure 5.6 shows before clicking the button and Figure 5.7 after clicking the button.

```java
Button button = (Button) findViewById(R.id.button);
final ImageView imageView = (ImageView)
findViewById(R.id.imageView);

button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        TransitionDrawable transitionDrawable = (TransitionDrawable)
        imageView.getDrawable();
        transitionDrawable.startTransition(1000);
    }
});
```

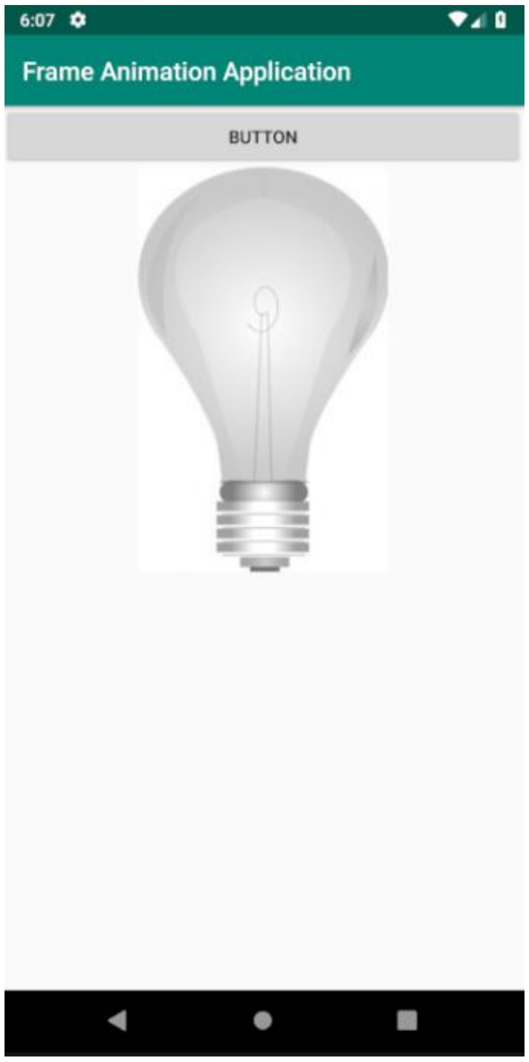


Figure 5.6 Before Clicking the Button

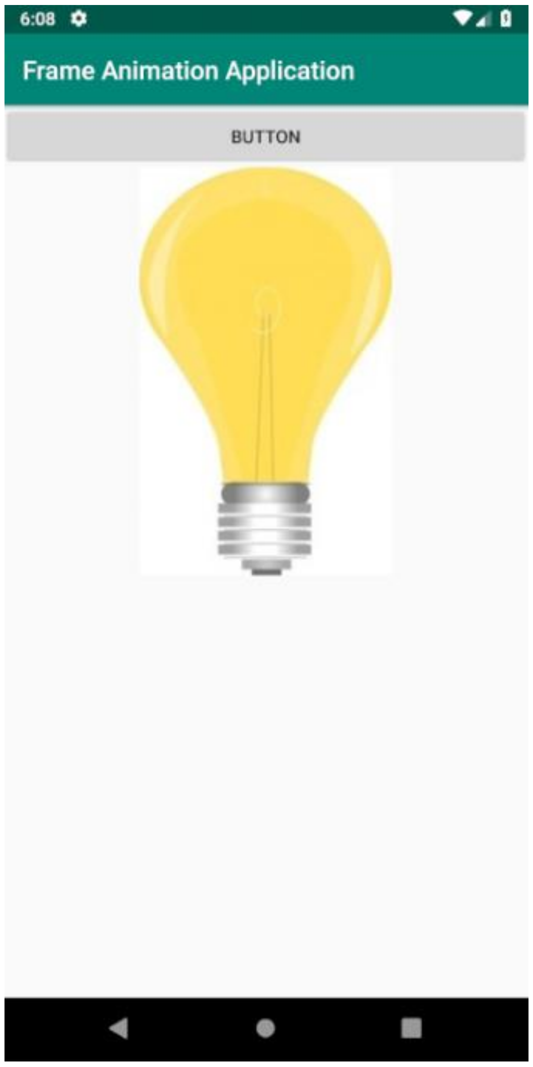


Figure 5.7 After Clicking the Button

- Try to change the method from startTransition to reverseTransition and understand the difference between these methods.

3.2. Tween animation Application

Create a new project with the name "Tween Animation Application". Tween Animation is defined as an animation which is used to Translate, Rotate, Scale and Alpha any type of view in Android.

- Add the image in Figure 5.8 to the res/drawable folder and call it image as shown in Figure 5.9.

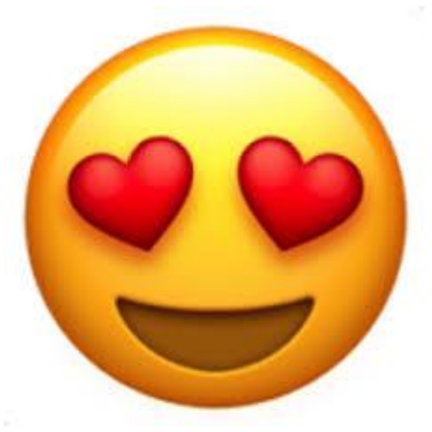
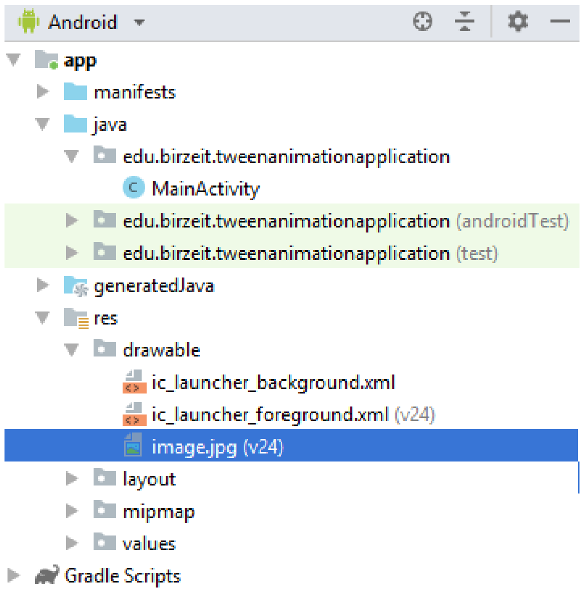


Figure 5.8 Tween Image



```
Android
▼ app
  ▶ manifests
  ▼ java
    ▼ edu.birzeit.tweenanimationapplication
        MainActivity
    ▶ edu.birzeit.tweenanimationapplication (androidTest)
    ▶ edu.birzeit.tweenanimationapplication (test)
  ▶ generatedJava
  ▼ res
    ▼ drawable
        ic_launcher_background.xml
        ic_launcher_foreground.xml (v24)
        image.jpg (v24)
    ▶ layout
    ▶ mipmap
    ▶ values
▶ Gradle Scripts
```

Figure 5.9 Showing drawable Directory

➢ In the res folder create a new Android Resource Directory as shown in Figure 5.10.

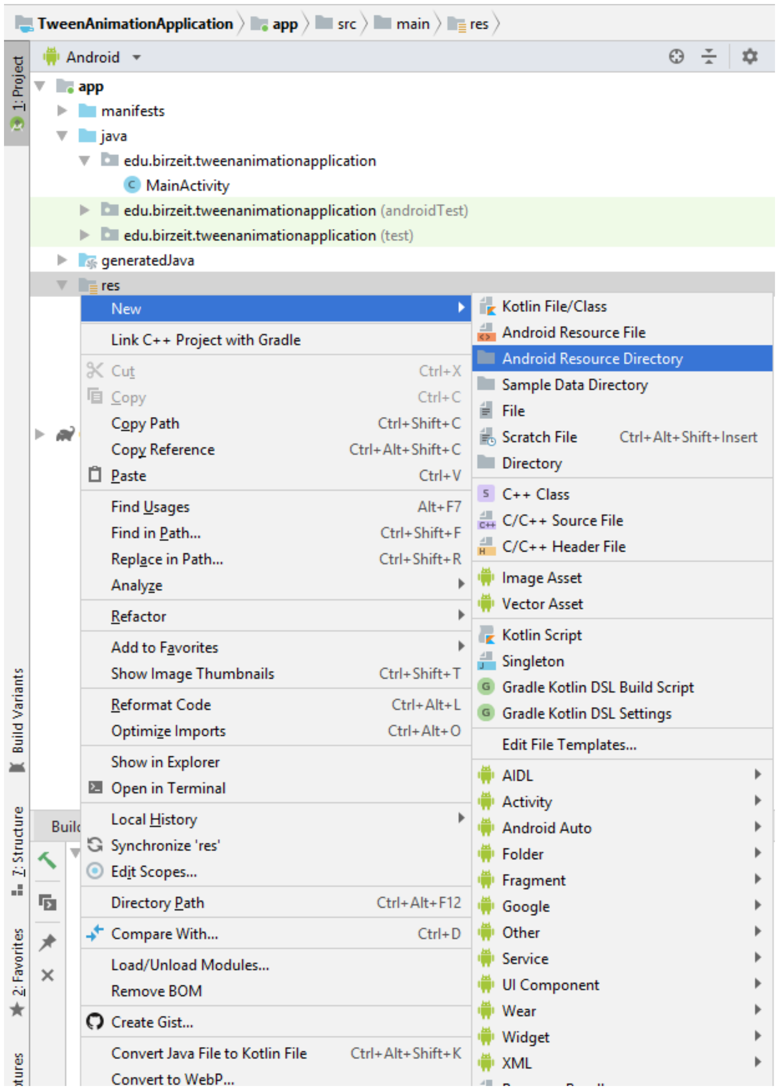
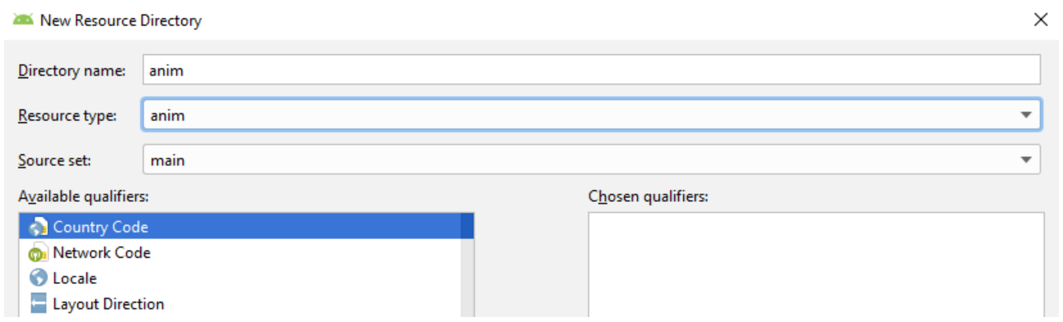


Figure 5.10 Creating Android Resource Directory Screen

➢ Change the Resource type to anim as show in Figure 5.11.


New Resource Directory

Directory name: anim

Resource type: anim

Source set: main

Available qualifiers:
- Country Code
- Network Code
- Locale
- Layout Direction

Chosen qualifiers:

Figure 5.11 New Resource Directory Screen

➢ Create three Animation Resource File in the anim directory and call them rotate, scale and translate as shown in Figure 5.12.

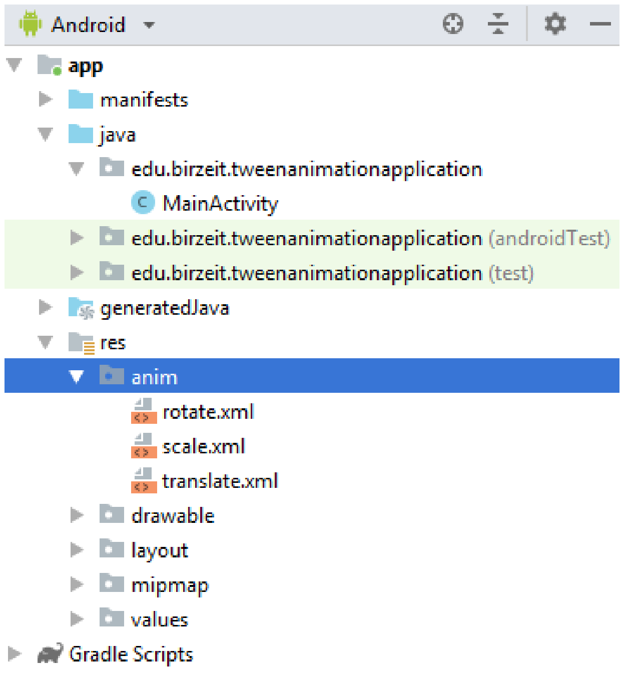
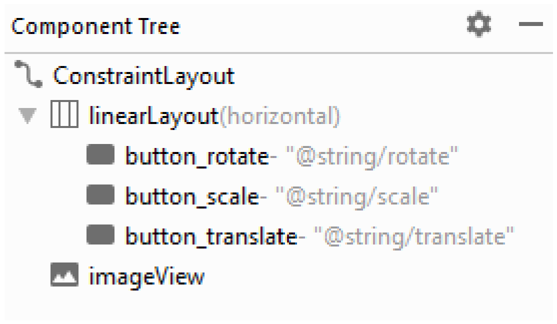

```
Android
app
  manifests
  java
    edu.birzeit.tweenanimationapplication
      MainActivity
    edu.birzeit.tweenanimationapplication (androidTest)
    edu.birzeit.tweenanimationapplication (test)
  generatedJava
  res
    anim
      rotate.xml
      scale.xml
      translate.xml
    drawable
    layout
    mipmap
    values
Gradle Scripts
```

Figure 5.12 showing anim Directory

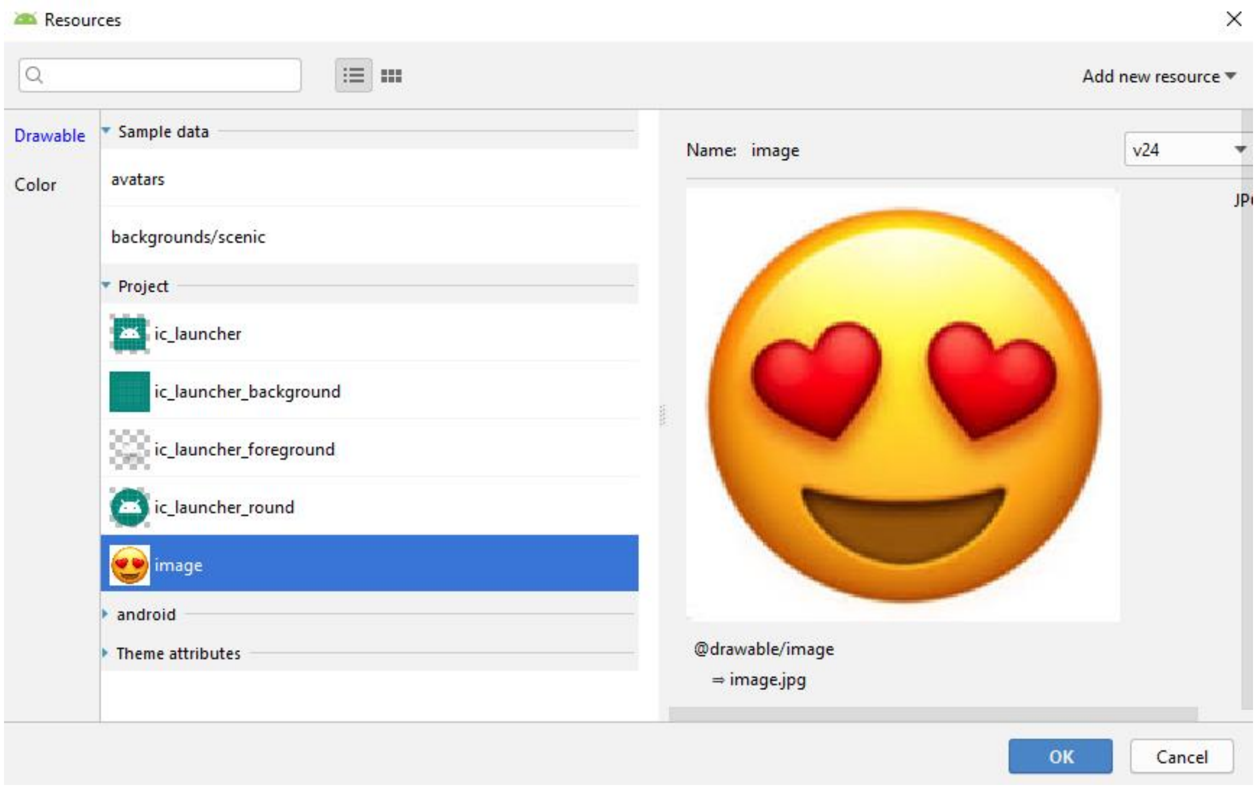
- ➢ In the activity_main layout makes the following design shown in Figure 5.13.



Component Tree
ConstraintLayout
linearLayout(horizontal)
button_rotate- "@string/rotate"
button_scale- "@string/scale"
button_translate- "@string/translate"
imageView

Figure 5.13 Component tree for Main Activity

- ➢ Add @drawable/image to the source compact when adding the image view as shown in Figure 5.14.



Resources

Add new resource

Drawable
Color

Sample data
avatars
backgrounds/scenic
Project
ic_launcher
ic_launcher_background
ic_launcher_foreground
ic_launcher_round
image
android
Theme attributes

Name: image
v24

@drawable/image
⇒ image.jpg

OK
Cancel

Figure 5.14 Adding the Image to the imageView

➢ Making the rotate animation (writing the code to the rotate.xml)

- Make the duration for the animation 3 seconds.
- Make the rotation 360 degree.
- Make the x coordination and the y coordination 50% (to itself).
- The following code is for the rotation.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:duration="3000"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toDegrees="360" />
</set>
```

➢ Making the Scale animation (writing the code to the scale.xml)

- Make the duration for the animation 2 seconds.
- Make the scale zoom in by making fromXScale=1, fromYScale=1 to toXScale=3, toYScale=3.
- Make the x coordination and the y coordination 50% (to itself).
- The following code is for the zoom in.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        android:duration="2000"
        android:fromXScale="1"
        android:fromYScale="1"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="3"
        android:toYScale="3"/>
</set>
```

➢ Making the Translate animation (writing the code to the translate.xml)

- Make the duration for the animation 3 seconds.
- Make the translate in the x axis from 0% to 30% (to its parent view).
- The following code is for the translate.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="3000"
        android:fromXDelta="0%p"
        android:toXDelta="30%p" />
</set>
```

➢ In the MainAvticivty java make 3 buttons by connecting them to the three buttons in the main activity layout.

➢ Make an ImageView object by connecting it the image view in the main activity layout.

➢ Make three setOnClickListeners to the three buttons and start the animation for each button by calling startAnimation method and passing the AnimationUtils xml folder.

➢ The code below shows how to make the animation for the three buttons and the imageView.

➢ The output is shown in

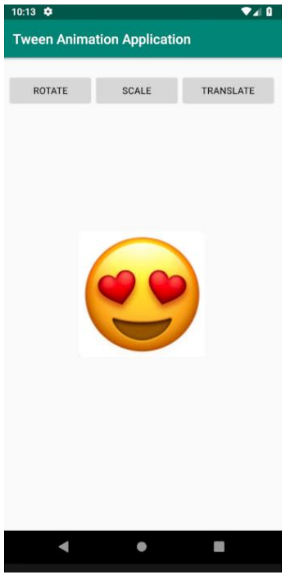


Figure 5.15 Tween Animation Application

```
Button buttonRotate=(Button)findViewById(R.id.button_rotate);
Button buttonScale=(Button) findViewById(R.id.button_scale);
Button buttonTranslate =(Button)
findViewById(R.id.button_translate);
final ImageView imageView = (ImageView)
findViewById(R.id.imageView);
buttonRotate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

imageView.startAnimation(AnimationUtils.loadAnimation(MainActivity.t
his,R.anim.rotate));
    }
});

buttonScale.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

imageView.startAnimation(AnimationUtils.loadAnimation(MainActivity.t
his,R.anim.scale));
    }
});

buttonTranslate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

imageView.startAnimation(AnimationUtils.loadAnimation(MainActivity.t
his,R.anim.translate));
    }
});
```

3.3. Try doing the following animations

- **(You must know what the following code does)**

  - The first code

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true">
    <alpha
        android:duration="1000"
        android:fromAlpha="0.0"
        android:toAlpha="1.0" />
</set>
```

➢ The second Code

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true"
    android:interpolator="@android:anim/linear_interpolator">
    <!-- Use startOffset to give delay between animations. It
specify the start time of the animation after you press the button
in that case -->
    <!-- Move -->
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromXDelta="0%p"
        android:startOffset="300"
        android:toXDelta="75%p" />
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromYDelta="0%p"
        android:startOffset="1100"
        android:toYDelta="70%p" />
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromXDelta="0%p"
        android:startOffset="1900"
        android:toXDelta="-75%p" />
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromYDelta="0%p"
        android:startOffset="2700"
        android:toYDelta="-70%p" />
    <!-- Rotate 360 degrees -->
    <rotate
        android:duration="1000"
        android:fromDegrees="0"
        android:interpolator="@android:anim/cycle_interpolator"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="infinite"
        android:repeatMode="restart"
        android:startOffset="3800"
        android:toDegrees="360" />
</set>
```

4. Todo

This part will be given to you by the teacher assistant in the lab time.