EX 5	Custom Views and Animations using Android's Canvas and Animator Classes

Aim

To create custom views and animations using Android's Canvas and Animator classes.

Definitions

Custom Views

In Android development, a custom view is a class that subclasses an existing `View` or `ViewGroup` to create a reusable user interface component with a unique appearance or behavior not available by default. They are essential for advanced UI design and performance optimization.

Animations

Animations in Android are UI techniques used to create motion, change view properties (size, position, transparency), and guide user interaction, significantly enhancing the app's visual appeal and user experience. The Android framework offers robust APIs—mainly **Property**, **View**, and **Drawable** animations—to transition between visual states smoothly.

Canvas Class

The `Canvas` class in Android is a fundamental component of the 2D drawing API, providing a drawing surface (like a "drawing board") where you can render custom graphics. It holds all the "draw" calls (e.g., `drawLine()`, `drawRect()`, `drawText()`, `drawBitmap()`) which are used to define what appears on the screen.

Animator Class

The `android.animation.Animator` class is the basic superclass in Android's property animation system, providing the fundamental structure and timing engine for creating animations.

Procedure

1. Open Android Studio IDE → go to File → New → New Project → specify the application name "CanvaAnimation" and company domain "com.mad.canani" → click "next" → choose Minimum SDK "API 17:Android 4.2(Jelly Bean)" → click "Next" → choose "Blank Activity" → click "next" → specify the Activity Name "MainActivity" → click "Finish".
2. Open MainActivity.java under app/java/ canani.mad.com.canvaanimation and type the following codes:

MainActivity.java

```
package canani.mad.com.canvaanimation;

import android.animation.AnimatorSet;
import android.animation.ArgbEvaluator;
import android.animation.ValueAnimator;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

   AnimatedCircleView circleView;

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

      circleView = (AnimatedCircleView) findViewById(R.id.circleView);
      Button btnAnimate = (Button) findViewById(R.id.btnAnimate);

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

   private void startAnimation() {

      // Radius animation
      ValueAnimator radiusAnimator = ValueAnimator.ofFloat(50f, 150f);
      radiusAnimator.setDuration(1000);
      radiusAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
         @Override
```

```
        public void onAnimationUpdate(ValueAnimator animation) {
            float value = (float) animation.getAnimatedValue();
            circleView.setRadius(value);
        }
    });

    // Color animation
    ValueAnimator colorAnimator = ValueAnimator.ofObject(
            new ArgbEvaluator(),
            Color.BLUE,
            Color.RED
    );
    colorAnimator.setDuration(1000);
    colorAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
        @Override
        public void onAnimationUpdate(ValueAnimator animation) {
            int color = (int) animation.getAnimatedValue();
            circleView.setCircleColor(color);
        }
    });

    // Play animations together
    AnimatorSet set = new AnimatorSet();
    set.playTogether(radiusAnimator, colorAnimator);
    set.start();
  }
}
```

3. Right click on canani.mad.com.canvaanimation package → New → Java Class → specify the class name as AnimatedCircleView → ok.
4.
5. Open AnimatedCircleView.java under app/java/ canani.mad.com.canvaanimation and type the following codes:

AnimatedCircleView.java

```
package canani.mad.com.canvaanimation;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;

public class AnimatedCircleView extends View {

  private Paint paint;
  private float radius = 50f;
```

```
    public AnimatedCircleView(Context context) {
        super(context);
        init();
    }

    public AnimatedCircleView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public AnimatedCircleView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init() {
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(Color.BLUE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        float cx = getWidth() / 2f;
        float cy = getHeight() / 2f;

        canvas.drawCircle(cx, cy, radius, paint);
    }

    public void setRadius(float radius) {
        this.radius = radius;
        invalidate(); // redraw view
    }

    public void setCircleColor(int color) {
        paint.setColor(color);
        invalidate();
    }
}
```

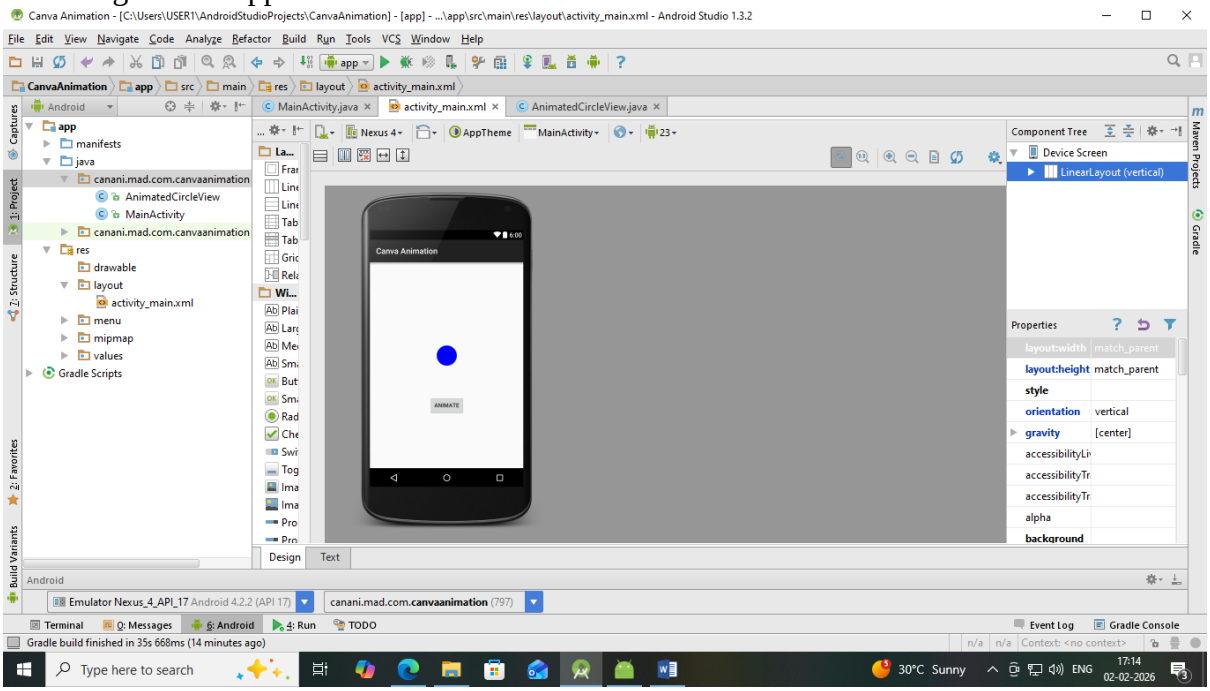
6. Open activity_main.xml under app/res/layout and type the following codes:

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
```

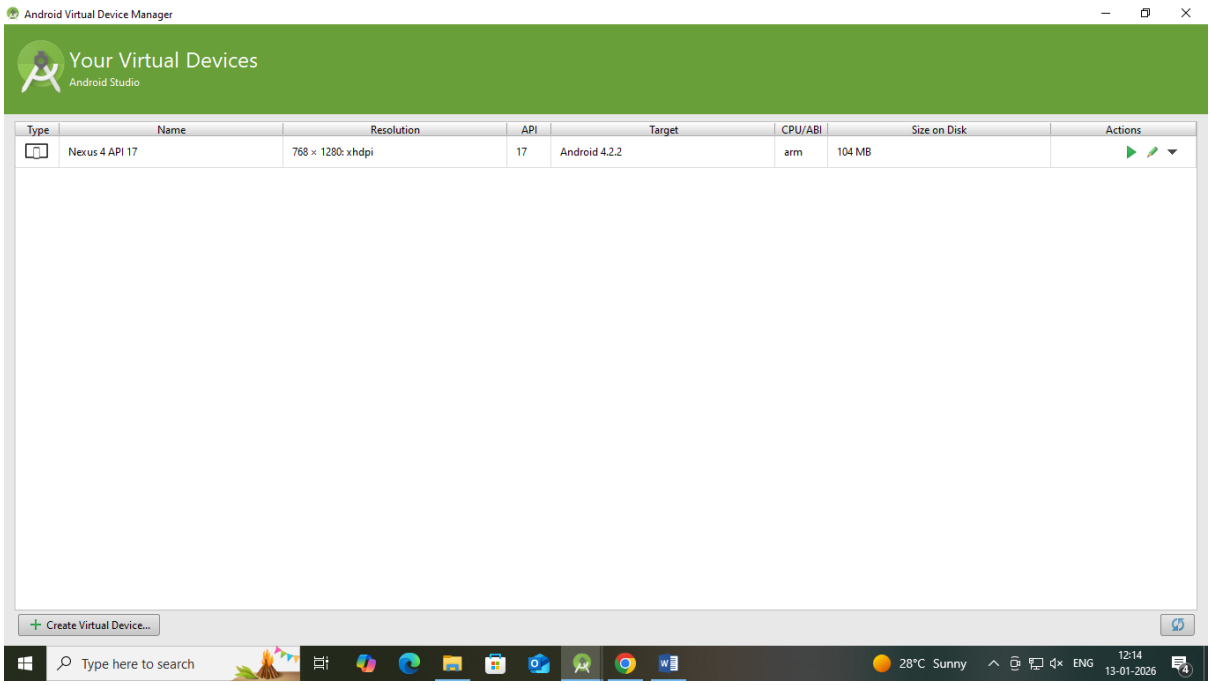
```
    android:orientation="vertical">

    <canani.mad.com.canvaanimation.AnimatedCircleView
        android:id="@+id/circleView"
        android:layout_width="200dp"
        android:layout_height="200dp" />

    <Button
        android:id="@+id/btnAnimate"
        android:text="Animate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

7. The design of the application will be as follows:



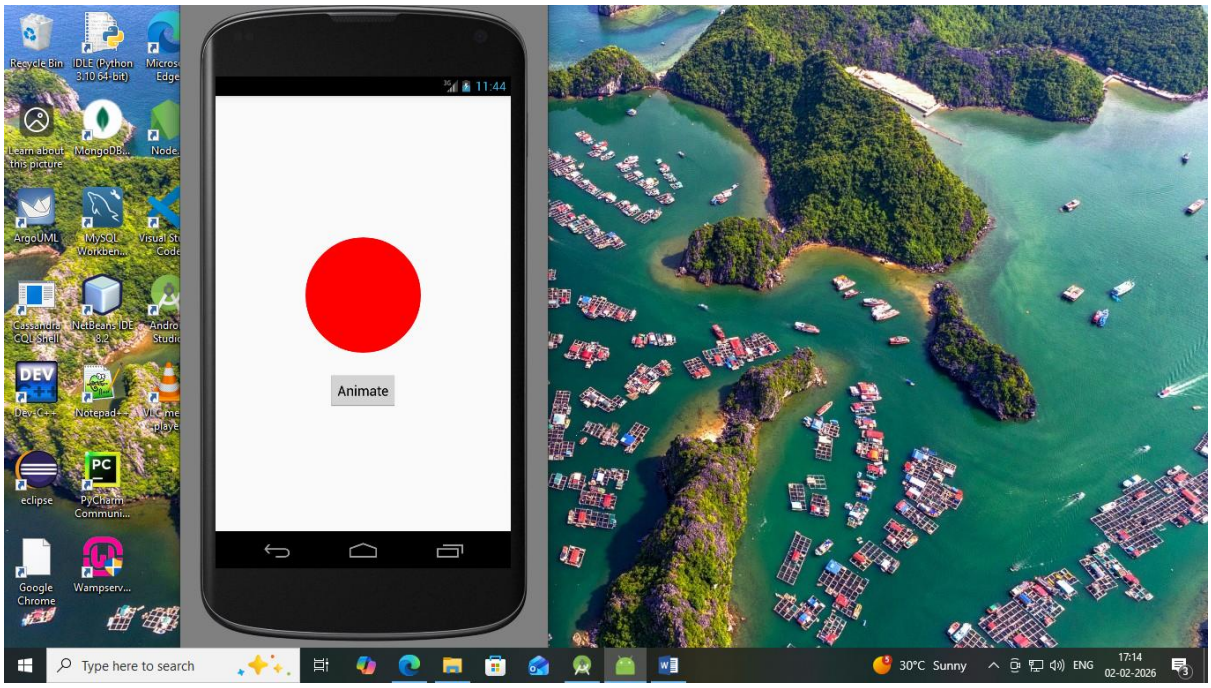
8. Go to Tools → android → AVD Manager → click "+ create a virtual device" → select "phone" from category → select "Nexus 4" from the list → click "next" → select Release name: Jelly Bean, API Level: 17, ABI: armeabi-v7a, Target: Android 4.2.2 from the list → click "next" → Choose orientation "portrait" → click "finish".

The following window will appear after configuring AVD:



9. Click “Run app” button in the Android Studio → choose android virtual device → click “ok”.

Output



Result

Thus, custom views and animations using Android's Canvas and Animator classes have been created.