Practical No. 7

**Title: Android program to work with graphics and animation Aim: Create an application to demonstrate graphics and animation Introduction**

# Create Drawing Objects

The [android.graphics](https://developer.android.com/reference/android/graphics/package-summary) framework divides drawing into two areas:

1. *What* to draw, handled by [Canvas](https://developer.android.com/reference/android/graphics/Canvas)
2. *How* to draw, handled by [Paint.](https://developer.android.com/reference/android/graphics/Paint)

For instance, [Canvas](https://developer.android.com/reference/android/graphics/Canvas) provides a method to draw a line, while [Paint](https://developer.android.com/reference/android/graphics/Paint) provides methods to define that line's color. [Canvas](https://developer.android.com/reference/android/graphics/Canvas) has a method to draw a rectangle, while [Paint](https://developer.android.com/reference/android/graphics/Paint) defines whether to fill that rectangle with a color or leave it empty. Simply put, [Canvas](https://developer.android.com/reference/android/graphics/Canvas) defines shapes that you can draw on the screen, while [Paint](https://developer.android.com/reference/android/graphics/Paint) defines the color, style, font, and so forth of each shape you draw.

# Draw!

Once you have your object creation and measuring code defined, you can implement [onDraw().](https://developer.android.com/reference/android/view/View#onDraw(android.graphics.Canvas)) Every view implements [onDraw()](https://developer.android.com/reference/android/view/View#onDraw(android.graphics.Canvas)) differently, but there are some common operations that most views share:

* Draw text using [drawText().](https://developer.android.com/reference/android/graphics/Canvas#drawText(char%5B%5D%2C%20int%2C%20int%2C%20float%2C%20float%2C%20android.graphics.Paint)) Specify the typeface by calling [setTypeface(),](https://developer.android.com/reference/android/graphics/Paint#setTypeface(android.graphics.Typeface)) and the text color by calling [setColor().](https://developer.android.com/reference/android/graphics/Paint#setColor(int))
* Draw primitive shapes using [drawRect(),](https://developer.android.com/reference/android/graphics/Canvas#drawRect(android.graphics.Rect%2C%20android.graphics.Paint)) [drawOval(),](https://developer.android.com/reference/android/graphics/Canvas#drawOval(android.graphics.RectF%2C%20android.graphics.Paint)) and [drawArc().](https://developer.android.com/reference/android/graphics/Canvas#drawArc(android.graphics.RectF%2C%20float%2C%20float%2C%20boolean%2C%20android.graphics.Paint)) Change whether the shapes are filled, outlined, or both by calling [setStyle().](https://developer.android.com/reference/android/graphics/Paint#setStyle(android.graphics.Paint.Style))
* Draw more complex shapes using the [Path](https://developer.android.com/reference/android/graphics/Path) class. Define a shape by adding lines and curves to a [Path](https://developer.android.com/reference/android/graphics/Path) object, then draw the shape using [drawPath().](https://developer.android.com/reference/android/graphics/Canvas#drawPath(android.graphics.Path%2C%20android.graphics.Paint)) Just as with primitive shapes, paths can be outlined, filled, or both, depending on the [setStyle().](https://developer.android.com/reference/android/graphics/Paint#setStyle(android.graphics.Paint.Style))
* Define gradient fills by creating [LinearGradient](https://developer.android.com/reference/android/graphics/LinearGradient) objects. Call [setShader()](https://developer.android.com/reference/android/graphics/Paint#setShader(android.graphics.Shader)) to use your [LinearGradient](https://developer.android.com/reference/android/graphics/LinearGradient) on filled shapes.
* Draw bitmaps using [drawBitmap().](https://developer.android.com/reference/android/graphics/Canvas#drawBitmap(android.graphics.Bitmap%2C%20android.graphics.Matrix%2C%20android.graphics.Paint))

**Exercise - Create android application to demonstrate graphics and animation**

# Implementation:

**Program: MainActivity.java**

import android.app.Activity;

import android.content.Context; import android.graphics.Canvas; import android.graphics.Color; import android.graphics.Paint; import android.os.Bundle; import android.view.View;

public class MainActivity extends Activity

{

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState); setContentView(new MyView(this));

}

public class MyView extends View

{

Paint paint = null;

public MyView(Context context)

{

super(context); paint = new Paint();}

@Override

protected void onDraw(Canvas canvas)

{

super.onDraw(canvas); int x = getWidth(); int y = getHeight(); int radius;

radius = 100; paint.setStyle(Paint.Style.FILL); paint.setColor(Color.WHITE); canvas.drawPaint(paint);

// Use Color.parseColor to define HTML colors paint.setColor(Color.parseColor("#CD5C5C")); canvas.drawCircle(x / 2, y / 2, radius, paint);

}

}

}

# Output:

