**Ex. No. : 03 Date :**

**Register No. : 221701043**

**Name : PUGAZHENTHI S**

**Graphical Primitives**

**Aim**

Develop an android application to draw the circle, ellipse, rectangle and some text using Android Graphical primitives.

***Procedure:***

**Step 1 :** File → New Project  
 Provide the application name (e.g.,"GraphicalPrimitives") and click “Next”.

**Step 2 :** Select the target Android devices  
 Select the minimum SDK to run the application. Click “Next”.

**Step 3 :** Choose the activity for the application  
 By default, choose “Blank Activity”. Click “Next”.

**Step 4 :** Enter activity name and click “Finish”.

**Step 5 :** Edit the program

Design Shapes and graphical elements in activity\_main.xml or

use Canvas API in kotlin code MainActivity.kt.  
**Step 6 :** Run the application  
 Two ways to run the application:

1. Running through emulator  
   
 2. Running through mobile device

***AndroidManifest.xml***

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

**<manifest xmlns:android="http://schemas.android.com/apk/res/android"**

**package="com.example.graphical primitives">**

**<application**

**android:allowBackup="true"**

**android:icon="@mipmap/ic\_launcher"**

**android:label="@string/app\_name"**

**android:roundIcon="@mipmap/ic\_launcher\_round"**

**android:supportsRtl="true"**

**android:theme="@style/Theme.GraphicalPrimitives">**

**<activity**

**android:name=".MainActivity"**

**android:exported="true">**

**<intent-filter>**

**<action android:name="android.intent.action.MAIN" />**

**<category android:name="android.intent.category.LAUNCHER" />**

**</intent-filter>**

**</activity>**

**</application>**

**</manifest>**

***Activity\_main.xml***

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

**<LinearLayout 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"**

**android:orientation="vertical"**

**tools:context=".MainActivity">**

**<TextView**

**android:layout\_width="match\_parent"**

**android:layout\_height="wrap\_content"**

**android:background="#673AB7"**

**android:padding="16dp"**

**android:text="Graphical Primitives"**

**android:textColor="#FFFFFF"**

**android:textSize="18sp"**

**android:textStyle="bold" />**

**<com.example.graphicalprimitives.ShapesView**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent"**

**android:background="#FFFF00" />**

**</LinearLayout>**

***MainActivity.kt***

**package com.example.graphical primitives**

**import android.os.Bundle**

**import android.appcompat.app.AppCompatActivity**

**class MainActivity : AppCompatActivity() {**

**override fun onCreate(savedInstanceState: Bundle?) {**

**super.onCreate(savedInstanceState)**

**setContentView(R.layout.*activity\_main*)**

**}**

**}**

***ShapesView.kt***

**package com.example.graphical primitives**

**import android.content.Context**

**import android.graphics.Canvas**

**import android.graphics.Color**

**import android.graphics.Paint**

**import android.graphics.RectF**

**import android.util.AttributeSet**

**import android.view.View**

**class ShapesView @JvmOverloads constructor(**

**context: Context,**

**attrs: AttributeSet? = null,**

**defStyleAttr: Int = 0**

**) : View(context, attrs, defStyleAttr) {**

***// Paints for different shapes***

**private val circlePaint = Paint().*apply* {**

***color* = Color.*RED***

***style* = Paint.Style.*FILL***

***isAntiAlias* = true**

**}**

**private val rectanglePaint = Paint().*apply* {**

***color* = Color.*GREEN***

***style* = Paint.Style.*FILL***

***isAntiAlias* = true**

**}**

**private val squarePaint = Paint().*apply* {**

***color* = Color.*BLUE***

***style* = Paint.Style.*FILL***

***isAntiAlias* = true**

**}**

**private val linePaint = Paint().*apply* {**

***color* = Color.*BLACK***

***style* = Paint.Style.*STROKE***

***strokeWidth* = 5f**

***isAntiAlias* = true**

**}**

**private val ellipsePaint = Paint().*apply* {**

***color* = Color.rgb(255, 165, 0) *// Orange***

***style* = Paint.Style.*FILL***

***isAntiAlias* = true**

**}**

**private val textPaint = Paint().*apply* {**

***color* = Color.*BLACK***

***textSize* = 40f**

***isAntiAlias* = true**

**}**

**override fun onDraw(canvas: Canvas) {**

**super.onDraw(canvas)**

**val width = *width*.toFloat()**

**val height = *height*.toFloat()**

***// Calculate grid dimensions***

**val cellWidth = width / 2**

**val cellHeight = height / 3**

***// Draw Circle (top-left cell)***

**val circleRadius = cellWidth / 4**

**canvas.drawCircle(**

**cellWidth / 2, *// x-coordinate of center***

**cellHeight / 2, *// y-coordinate of center***

**circleRadius, *// radius***

**circlePaint**

**)**

***// Draw label for Circle***

**canvas.drawText(**

**"Circle",**

**cellWidth / 2 - 50f,**

**cellHeight / 5,**

**textPaint**

**)**

***// Draw Rectangle (top-right cell)***

**val rectLeft = cellWidth + cellWidth / 4**

**val rectTop = cellHeight / 4**

**val rectRight = cellWidth + 3 \* cellWidth / 4**

**val rectBottom = 3 \* cellHeight / 4**

**canvas.drawRect(**

**rectLeft,**

**rectTop,**

**rectRight,**

**rectBottom,**

**rectanglePaint**

**)**

***// Draw label for Rectangle***

**canvas.drawText(**

**"Rectangle",**

**cellWidth + cellWidth / 2 - 80f,**

**cellHeight / 5,**

**textPaint**

**)**

***// Draw Square (middle-left cell)***

**val squareSize = cellWidth / 2**

**val squareLeft = cellWidth / 4**

**val squareTop = cellHeight + cellHeight / 4**

**canvas.drawRect(**

**squareLeft,**

**squareTop,**

**squareLeft + squareSize,**

**squareTop + squareSize,**

**squarePaint**

**)**

***// Draw label for Square***

**canvas.drawText(**

**"Square",**

**cellWidth / 2 - 50f,**

**cellHeight + cellHeight / 5,**

**textPaint**

**)**

***// Draw Line (middle-right cell)***

**val lineStartX = cellWidth + cellWidth / 4**

**val lineStartY = cellHeight + cellHeight / 2**

**val lineEndX = cellWidth + 3 \* cellWidth / 4**

**val lineEndY = cellHeight + cellHeight / 2**

**canvas.drawLine(**

**lineStartX,**

**lineStartY,**

**lineEndX,**

**lineEndY,**

**linePaint**

**)**

***// Draw label for Line***

**canvas.drawText(**

**"Line",**

**cellWidth + cellWidth / 2 - 40f,**

**cellHeight + cellHeight / 5,**

**textPaint**

**)**

***// Draw Ellipse (bottom-left cell)***

**val ellipseRect = RectF(**

**cellWidth / 4,**

**2 \* cellHeight + cellHeight / 4,**

**3 \* cellWidth / 4,**

**2 \* cellHeight + 3 \* cellHeight / 4**

**)**

**canvas.drawOval(ellipseRect, ellipsePaint)**

***// Draw label for Ellipse***

**canvas.drawText(**

**"Ellipse",**

**cellWidth / 2 - 50f,**

**2 \* cellHeight + cellHeight / 5,**

**textPaint**

**)**

***// Draw Text demo (bottom-right cell)***

**canvas.drawText(**

**"Sample Text",**

**cellWidth + cellWidth / 2 - 90f,**

**2 \* cellHeight + cellHeight / 2,**

**textPaint**

**)**

***// Draw label for Text***

**canvas.drawText(**

**"Text",**

**cellWidth + cellWidth / 2 - 40f,**

**2 \* cellHeight + cellHeight / 5,**

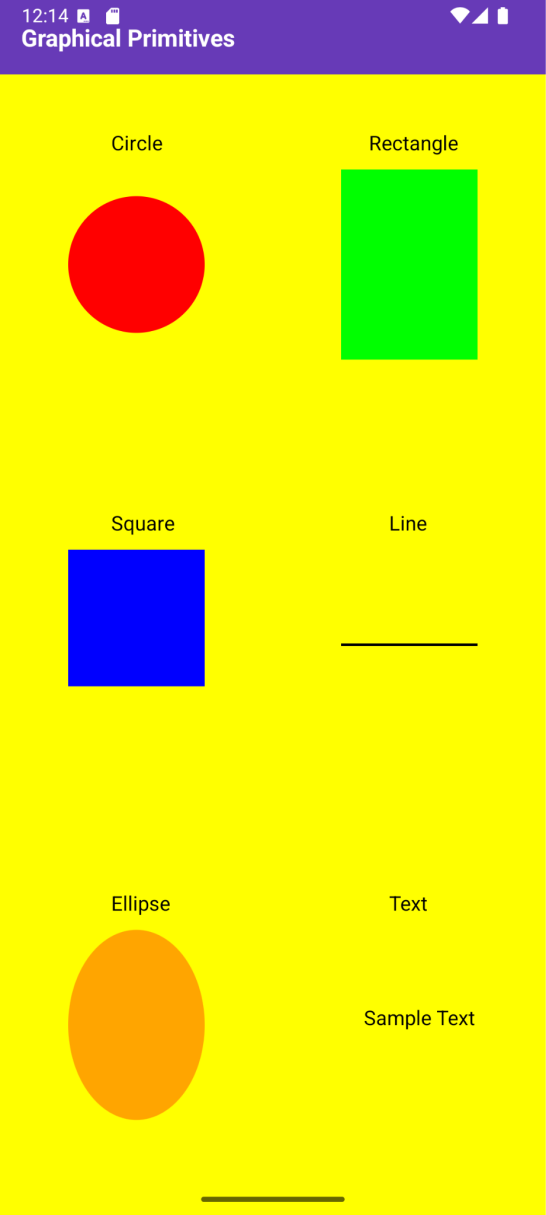
**textPaint**

**)**

**}**

**}**

***Output***

******

**Result:**

The Graphical Primitives application successfully displays shapes and graphical elements using Kotlin’s drawing functions when run on an emulator or mobile device.