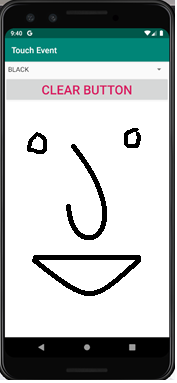
PRACTICAL 6

Part A

**AIM:** Implementing touch event.

**Scenario:** Create an application to draw line on the screen as the user drags his/her finger on the screen.

****

**Part B (to be completed by students)**

**(Students must submit the soft copy as per the following segments. The soft copy must be uploaded on the Blackboard. The filename should be Batch\_RollNo\_Exp\_No)**

|  |  |
| --- | --- |
| **Roll No.: K005** | **Name: Jal bafana** |
| **Prog/Yr/Sem: Cyber Sec/2nd/ Sem-4** | **Batch: K1** |
| **Date of Experiment: 02.04.2025** | **Date of Submission: 04.04.2025** |

1. **Program Scenario and Program code:** (Write Scenario and Paste your program code (Java, xml resource and layout)).

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <!-- Heading for the app -->  
 <TextView  
 android:id="@+id/heading"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Drawing App - Jal Bafana - K005"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 android:layout\_gravity="center"  
 android:paddingBottom="16dp"  
 android:textColor="#000000" />  
  
 <!-- Spinner for selecting pen color (full width) -->  
 <Spinner  
 android:id="@+id/colorSpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:entries="@array/colors\_array"  
 android:paddingBottom="16dp" />  
  
 <!-- Clear Button (full width) -->  
 <Button  
 android:id="@+id/clearButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Clear"  
 android:layout\_gravity="center"  
 android:paddingBottom="16dp" />  
  
 <!-- Custom Drawing View (drawing area takes the remaining space) -->  
 <com.example.k005\_lab6.DrawingView  
 android:id="@+id/drawingView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1" />  
  
</LinearLayout>

package com.example.k005\_lab6;  
  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.Spinner;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainActivity extends AppCompatActivity {  
  
 private DrawingView drawingView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 // Get references to the views  
 drawingView = findViewById(R.id.*drawingView*);  
 Spinner colorSpinner = findViewById(R.id.*colorSpinner*);  
 Button clearButton = findViewById(R.id.*clearButton*);  
  
 // Set up Spinner to handle color selection  
 colorSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id) {  
 // Change the drawing pen color based on the selected color  
 switch (position) {  
 case 0:  
 drawingView.setPenColor(0xFF000000); // Black  
 break;  
 case 1:  
 drawingView.setPenColor(0xFFFF0000); // Red  
 break;  
 case 2:  
 drawingView.setPenColor(0xFF0000FF); // Blue  
 break;  
 case 3:  
 drawingView.setPenColor(0xFF008000); // Green  
 break;  
 case 4:  
 drawingView.setPenColor(0xFFFFFF00); // Yellow  
 break;  
 }  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> parentView) {  
 // Default behavior when nothing is selected  
 }  
 });  
  
 // Set up the button click listener to clear the drawing  
 clearButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 drawingView.clear(); // Call the clear method to reset the drawing  
 }  
 });  
 }  
}

<resources>  
 <string name="app\_name">K005\_lab6</string>  
 <string-array name="colors\_array">  
 <item>Black</item>  
 <item>Red</item>  
 <item>Blue</item>  
 <item>Green</item>  
 <item>Yellow</item>  
 </string-array>  
</resources>

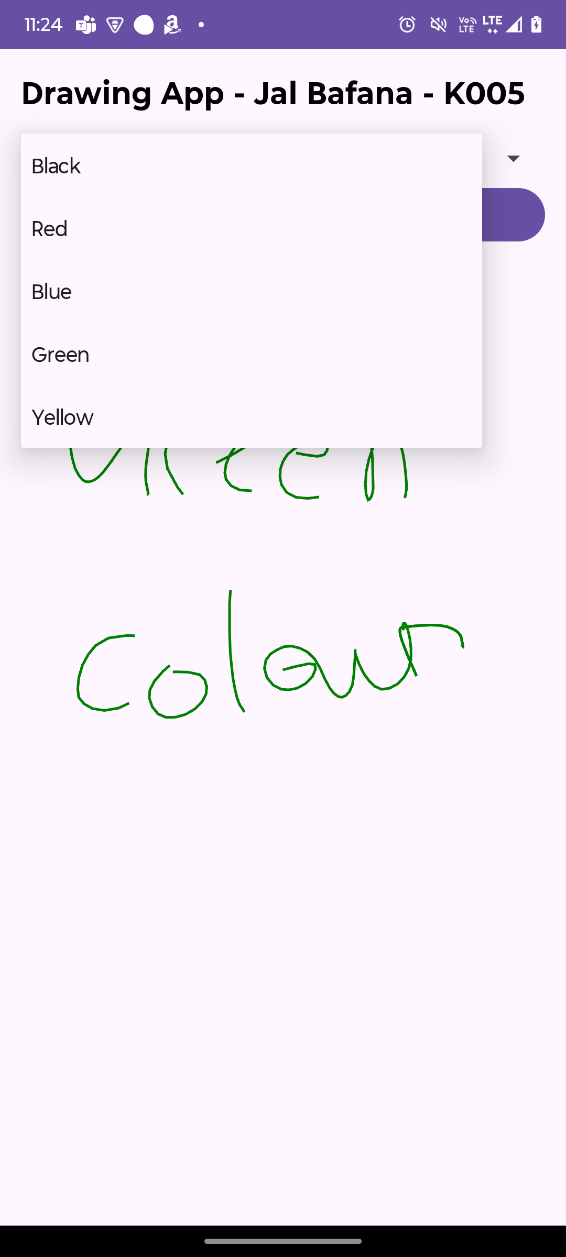
package com.example.k005\_lab6;  
  
import android.content.Context;  
import android.graphics.Canvas;  
import android.graphics.Paint;  
import android.graphics.Path;  
import android.util.AttributeSet;  
import android.view.MotionEvent;  
import android.view.View;  
import java.util.ArrayList;  
  
public class DrawingView extends View {  
 private ArrayList<Path> paths = new ArrayList<>(); // Store all paths drawn  
 private ArrayList<Paint> paints = new ArrayList<>(); // Store the paints used for each path  
 private Path currentPath;  
 private Paint currentPaint;  
  
 public DrawingView(Context context) {  
 super(context);  
 init();  
 }  
  
 public DrawingView(Context context, AttributeSet attrs) {  
 super(context, attrs);  
 init();  
 }  
  
 private void init() {  
 // Initialize the current path and paint  
 currentPath = new Path();  
 currentPaint = new Paint();  
 currentPaint.setColor(0xFF000000); // Default color (Black)  
 currentPaint.setStrokeWidth(5f); // Set the stroke width  
 currentPaint.setStyle(Paint.Style.*STROKE*); // Set the paint style to stroke  
 currentPaint.setAntiAlias(true); // Enable anti-aliasing  
 }  
  
 @Override  
 protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
 // Draw all the paths  
 for (int i = 0; i < paths.size(); i++) {  
 canvas.drawPath(paths.get(i), paints.get(i)); // Draw each path with its respective paint  
 }  
 // Draw the current path being drawn  
 canvas.drawPath(currentPath, currentPaint);  
 }  
  
 @Override  
 public boolean onTouchEvent(MotionEvent event) {  
 float x = event.getX();  
 float y = event.getY();  
  
 switch (event.getAction()) {  
 case MotionEvent.*ACTION\_DOWN*:  
 currentPath = new Path(); // Start a new path  
 currentPath.moveTo(x, y);  
 return true;  
 case MotionEvent.*ACTION\_MOVE*:  
 currentPath.lineTo(x, y); // Draw a line to the new touch point  
 break;  
 case MotionEvent.*ACTION\_UP*:  
 // When the user lifts their finger, save the current path and paint  
 paths.add(currentPath);  
 paints.add(currentPaint);  
 break;  
 default:  
 return false;  
 }  
 invalidate(); // Redraw the view  
 return true;  
 }  
  
 // Method to clear the drawing  
 public void clear() {  
 paths.clear(); // Clear the list of paths  
 paints.clear(); // Clear the list of paints  
 invalidate(); // Redraw the view  
 }  
  
 // Method to set the pen color  
 public void setPenColor(int color) {  
 currentPaint.setColor(color); // Change the paint color for the current path  
 }  
}

1. **Output:** (Paste your program input and output screen shots).

****

****

****

****

1. **Observations:** A brief description of the design aspects and working of the code in your own words.

* The app allows users to draw lines by dragging their fingers.
* A color selection spinner enables users to change the pen color dynamically.
* A "Clear" button resets the canvas.
* The DrawingView class manages touch events and rendering.

1. **Questions:** Draw & Explain with respect to layouts the Scene Graph of the experiment.

**LinearLayout**

**│── TextView (App Heading)**

**│── Spinner (Color Selector)**

**│── Button (Clear Canvas)**

**└── DrawingView (Custom Drawing Area)**

* Root Layout: LinearLayout (vertical orientation).
* TextView: Displays the app title.
* Spinner: Dropdown menu for selecting colors.
* Button: Clears the drawing area.
* Custom View (DrawingView): Handles touch input and rendering.

1. **Conclusion (Learning Outcomes):** How were the outcomes defined for the experiment in Part A fulfilled through the scenarios?

The experiment successfully implemented touch events for drawing, enabling dynamic user interaction. It enhanced understanding of custom views, event handling, and UI integration in Android.