TEST TOUCH

Submitted by

SRIJIB ROY

From Department of

MCA

TECHNO INDIA COLLEGE OF TECHNOLOGY

INDEX

SERIAL NUMBER	TOPIC
01	Introduction
02	Theoretical Background
03	Minimum System Requirement
04	Working Details
05	Use Case Diagram
06	Interface Design
07	Coding
08	Coding Standards Followed And Assumptions
09	Testing
10	Future Scope
11	References And Bibliography

INTRODUCTION

The “Test Touch” Application turns your device’s screen into a virtual canvas. We can paint by dragging fingers across the screen. The applications options enable us to set the drawing color, canvas color and line width. We can save the current drawing into our device’s Gallery (Save Image). At any point, we can shake the device to clear the entire drawing from the screen.

Our program uses the following features of android programming:-

- Two Dimensional Graphics
- Sensor Manager
- Use of Toasts
- Use of Canvas Class and save an image as bitmap

In this Application we implemented following characteristics:-

- Detect when the user touches the screen, moves a finger across the screen and removes a finger from the screen.
- Use a Sensor Manager to detect accelerometer motion events to clear the screen when the user shakes the device.
- Use a Paint object to specify the color and width of a line.
- Use Path objects to store each line’s data as the user draws the lines and to draw those lines with a Canvas.
- Use a Toast to briefly display a message on the screen.

THEORETICAL BACKGROUND

ANDROID BASED APPLICATION

This project has been developed by using android technology keeping in mind the rapid increase in the popularity of android devices. Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers.

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

HISTORY :

- Initially developed by Android, Inc., which Google backed financially and later bought in 2005.
- Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices.
- Android is an open source and so has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language. Therefore, Android has become the world's most widely used smart phone platform.
- **Open handset alliance(OHA):**

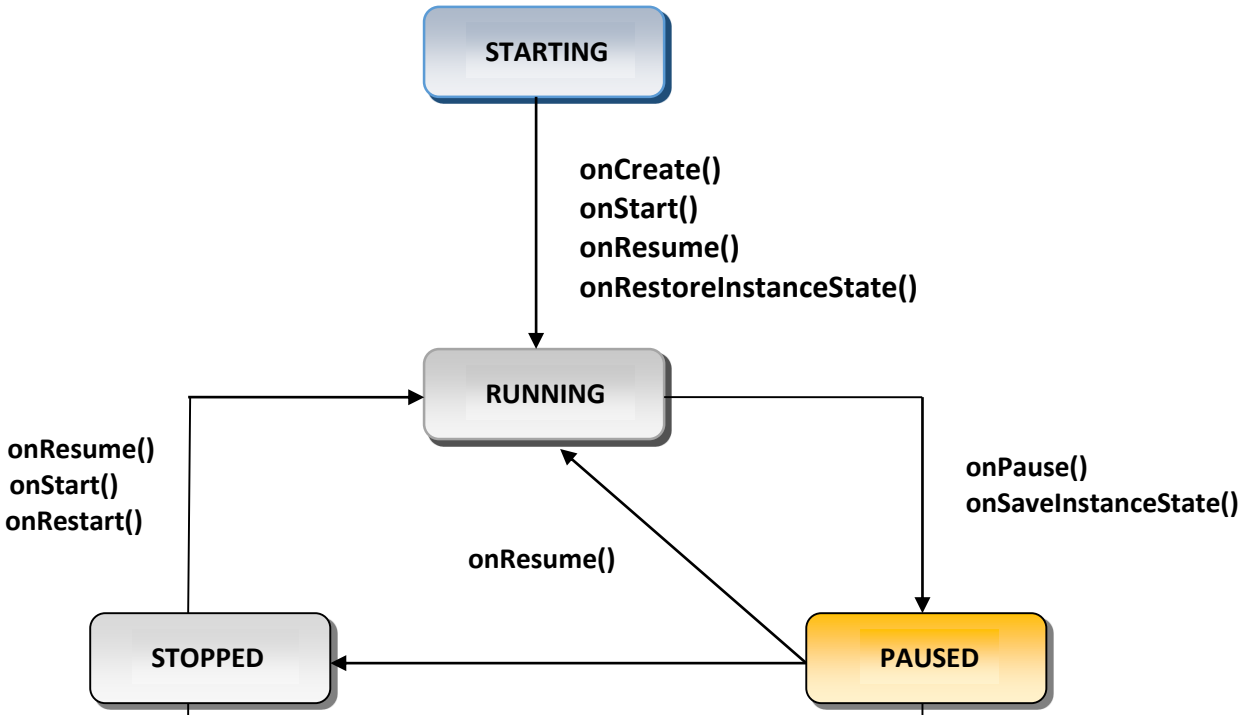
  - It is a consortium of 84 firms to develop open standards for mobile devices.
  - They have come together to accelerate innovations in mobiles and offer consumers a richer, less expensive and better mobile experience.

FEATURES:

- Application framework enabling reuse and replacement of components.
- Android, while recognizing and allowing for programmatic UI development, also supports the newer, XML-based UI layout.
- One of the more exciting and compelling features of Android is that, because of its architecture, third-party applications—including those that are “home grown”—are executed with the same system priority as those that are bundled with the core system.

- Each application is executed within its own thread using a very lightweight virtual machine called the Dalvik virtual machine optimized for mobile devices.
- Android include an accelerated 3-D graphics engine (based on hardware support), database support powered by SQLite, and an integrated web browser.
- Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.
- Aside from the very generous SDK and the well-formed libraries that are available to us to develop with, the most exciting feature for Android developers is that we now have access to anything the operating system has access to.
- Developers of Android applications will be able to tie their applications into existing Google offerings such as Google Maps and the omnipresent Google Search.
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE.

LIFE CYCLE:



During its lifetime, an android activity can be in any of the states as shown in the earlier diagram. Android will call the various methods in the Activity class at the appropriate times. The methods are :

- **onCreate(Bundle):** This is called when the activity first starts up. onCreate() takes only one parameter that is either null or some state information previously saved by the onSaveInstanceState() method.
- **onStart():** This indicates the activity is about to be displayed to the user.
- **onResume():** This is called when the activity can start interacting with the user.
- **onPause():** This runs when the activity is about to go into the background, usually because another activity has been launched.
- **onStop():** This is called when the activity is no longer visible to the user and it is not needed for a while.
- **onRestart():** If this method is called, it indicates the activity is redisplayed to the user from a stopped state.
- **onDestroy():** This is called right before the activity is destroyed.
- **onSaveInstanceState(Bundle):** Android will call this method to allow the activity to save pre-instance state.
- **onRestoreInstanceState(Bundle):** This is called when the activity is being reinitialized from a state previously saved by the onSaveInstanceState() method.

MINIMUM SYSTEM REQUIREMENT:

- **SOFTWARE REQUIREMENTS**

- Windows XP/ Windows 7 / Windows 8
- Java SE 6 Software Development Kit
- Eclipse IDE for Java Developers
- Android SDK (Software Development Kit)
- ADT (Android Development Tools) Plug-in for Eclipse

➢ HARDWARE REQUIREMENTS

- 2 GB RAM
- Display Unit
- Dual Core Processor with minimum 2GHZ clock speed
- 4GB of Free Space

WORKING DETAILS

- **Using SensorManager to Listen for Accelerometer Events :-**

This app allows the user to shake the device to erase the current drawing. Most devices have an accelerometer that allows apps to detect movement. Other sensors currently supported by Android include gravity, gyroscope, light, linear acceleration, magnetic field, pressure, proximity, rotation vector and temperature.

To listen for sensor events, you get a reference to the system's SensorManager service, which enables the app to receive data from the device's sensors. You use the SensorManager to register the sensor changes that your app should receive and to specify the SensorEventListener that will handle those sensor-change events. The classes and interfaces for processing sensor events are located in package android.hardware.

- **Creating Menu item Options:-**

Several apps use menu options to display information of several functions to the user or to ask questions and receive responses from the user in the form of Button/Menu clicks. In this app, we use these to allow the user to select a drawing color or select a line width or select a background/Canvas color.

- **Drawing Lines and Paths :-**

This app draws lines onto Bitmaps (package android. graphics). You can associate a Canvas with a Bitmap, and then use the Canvas to draw on the Bitmap, which can then be displayed on the screen. A Bitmap can also be saved into a file—we'll use this capability to store drawings in the device's gallery when the user touches the 'Save Image' menu item.

Processing Touch Events :-

The user can touch the screen with finger and drag the finger to draw line. We store the information for each individual finger as a Path object (package android.graphics), which represents a geometric path consisting of line segments and curves. Touch events are processed by overriding the View method OnTouchEvent. This method receives a MotionEvent (package android.view) that contains the type of touch event. We use the touch events to add information to the Path objects. We use the type of the touch event to determine whether the user has touched the screen, dragged across the screen or lifted a finger from the screen.

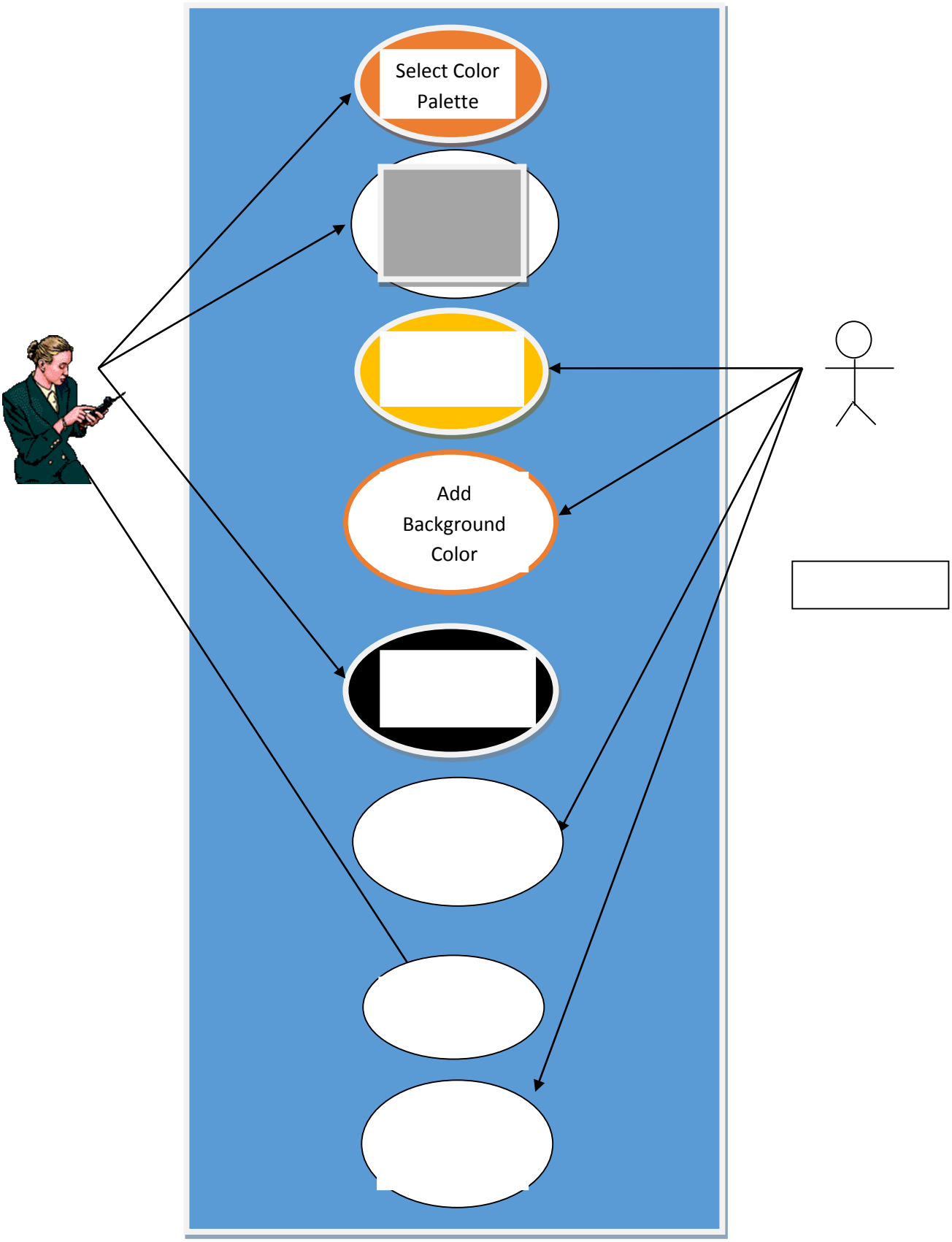
- **Saving the Drawing to the Device's Gallery :-**

The app provides a Save Image menu item that allows the user to save a drawing into the device's gallery—the default location in which photos taken with the device are stored. A Content Resolver (package android.content) enables the app to read data from and store data on a device. We'll use one to get an OutputStream for writing data into the gallery and save the image in JPEG format.

- **Using Toasts to Display a Message for a Short Time :-**

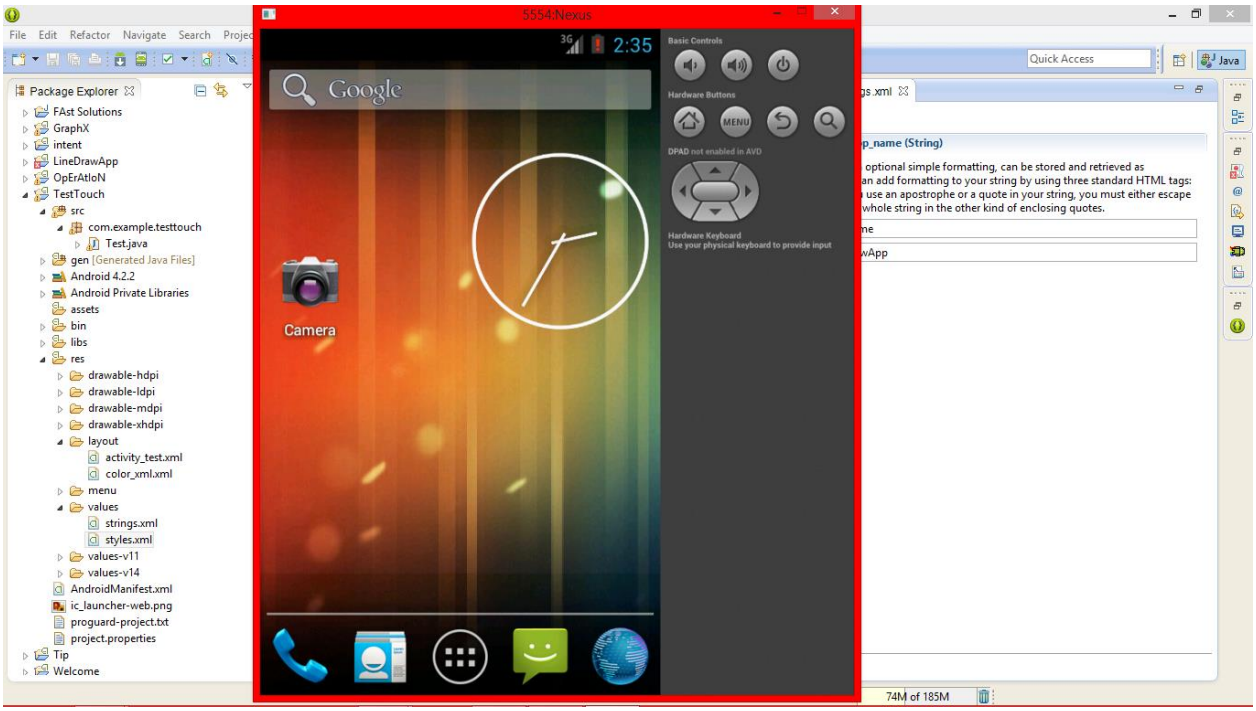
A Toast (package android.widget) displays a message for a short time, and then disappears from the screen. These are often used to display minor error messages or informational messages, such as an indication that indicates whether or not the user's drawing was successfully saved to the gallery.

USE CASE DIAGRAM

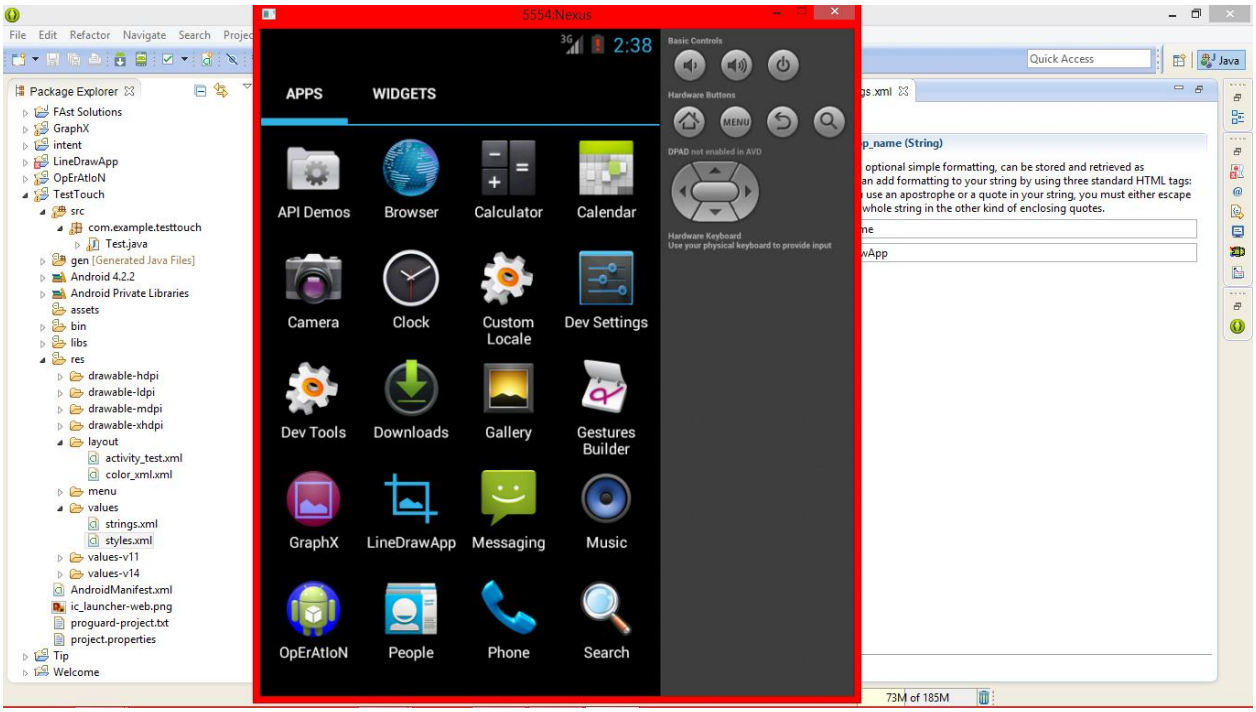


INTERFACE DESIGN

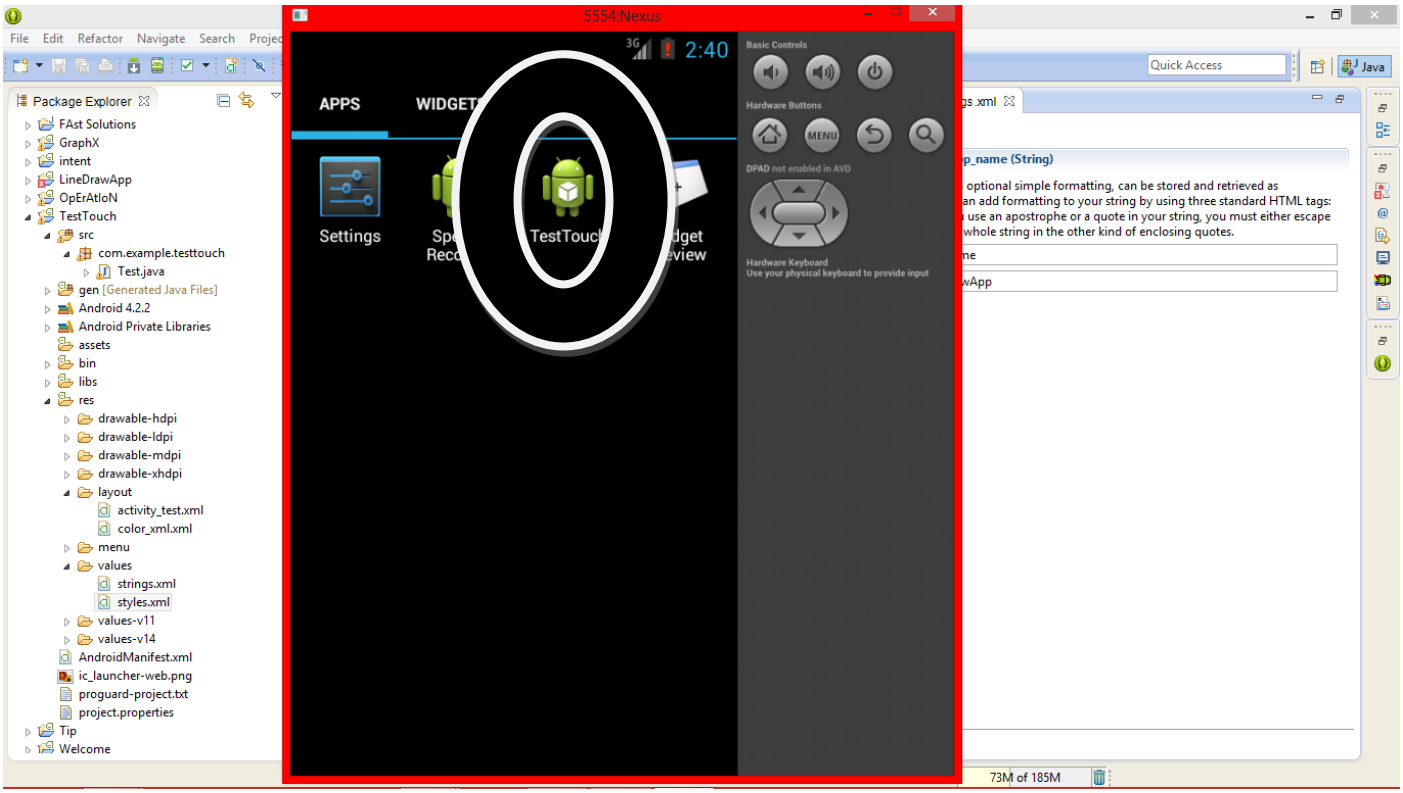
Screen Design 1: Overview of Android Emulator



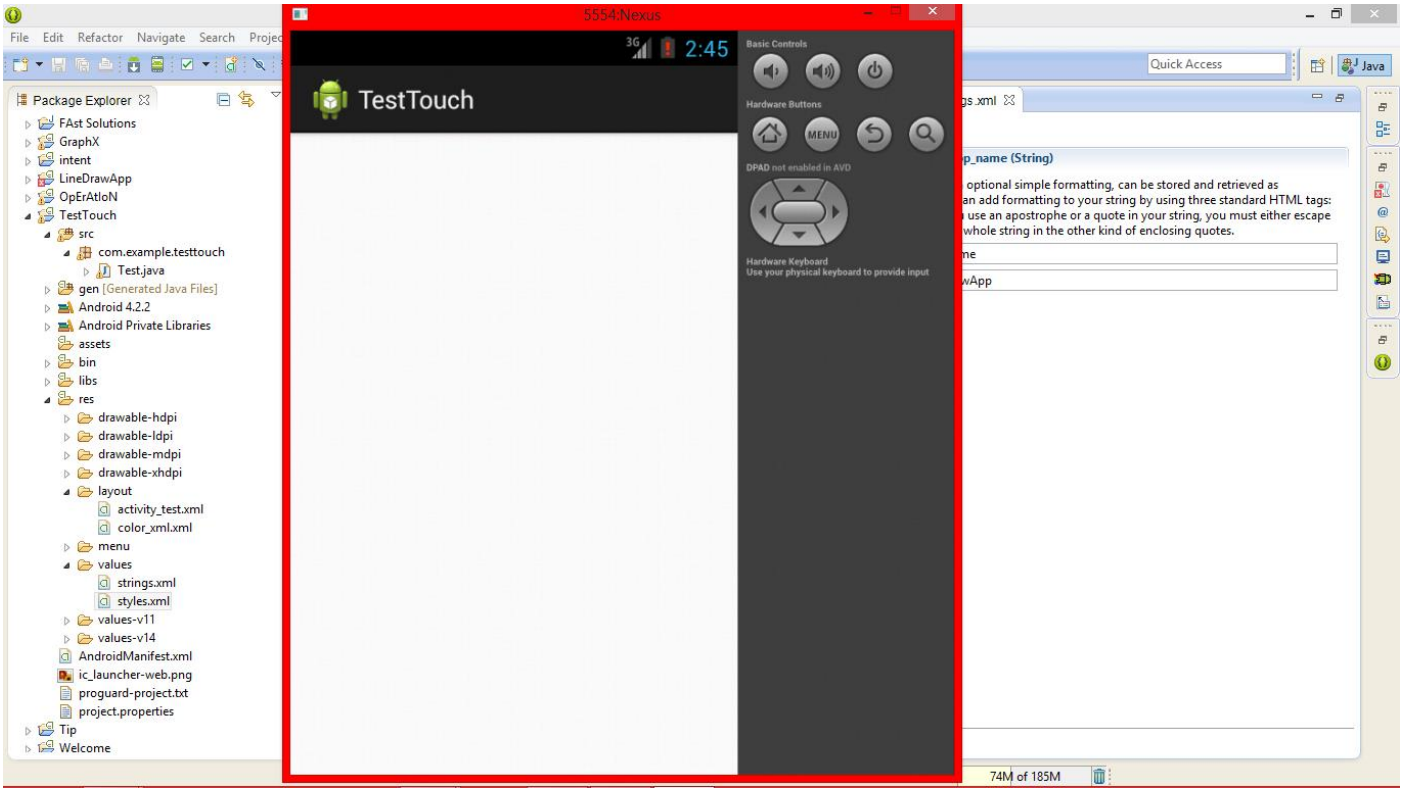
Screen Design 2: Application Overview Of Emulator



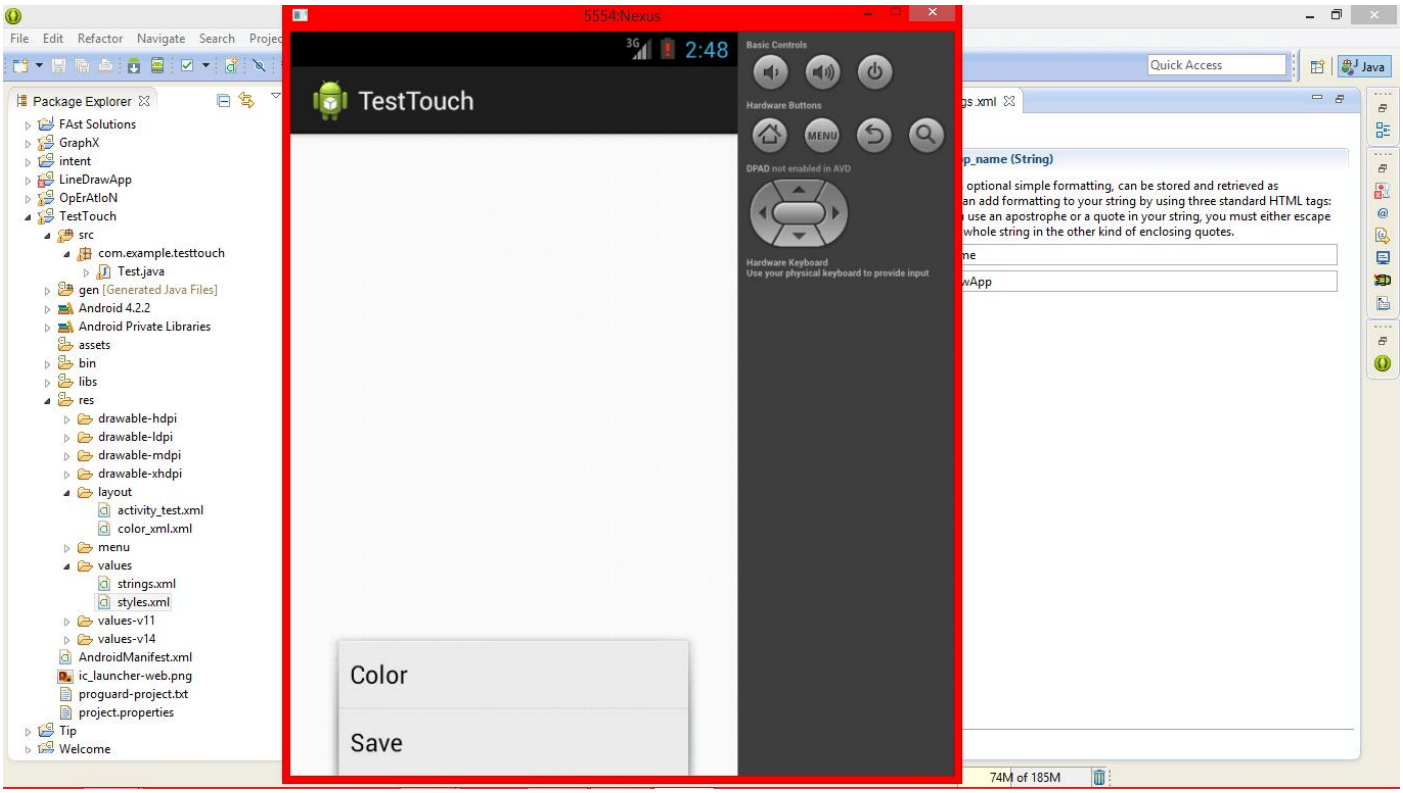
Screen Design 3: Android Application of Test Touch



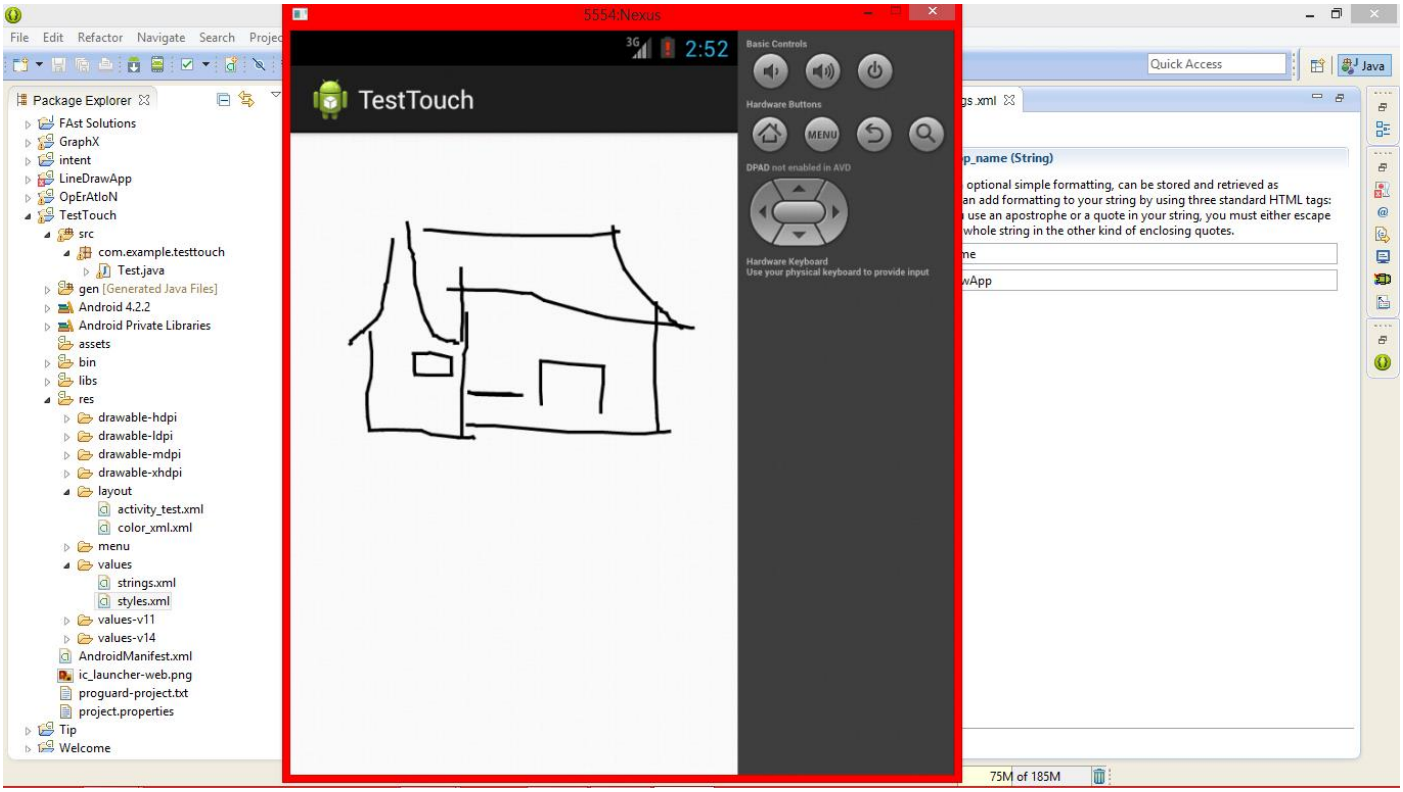
Screen Design 4: Drawing Canvas



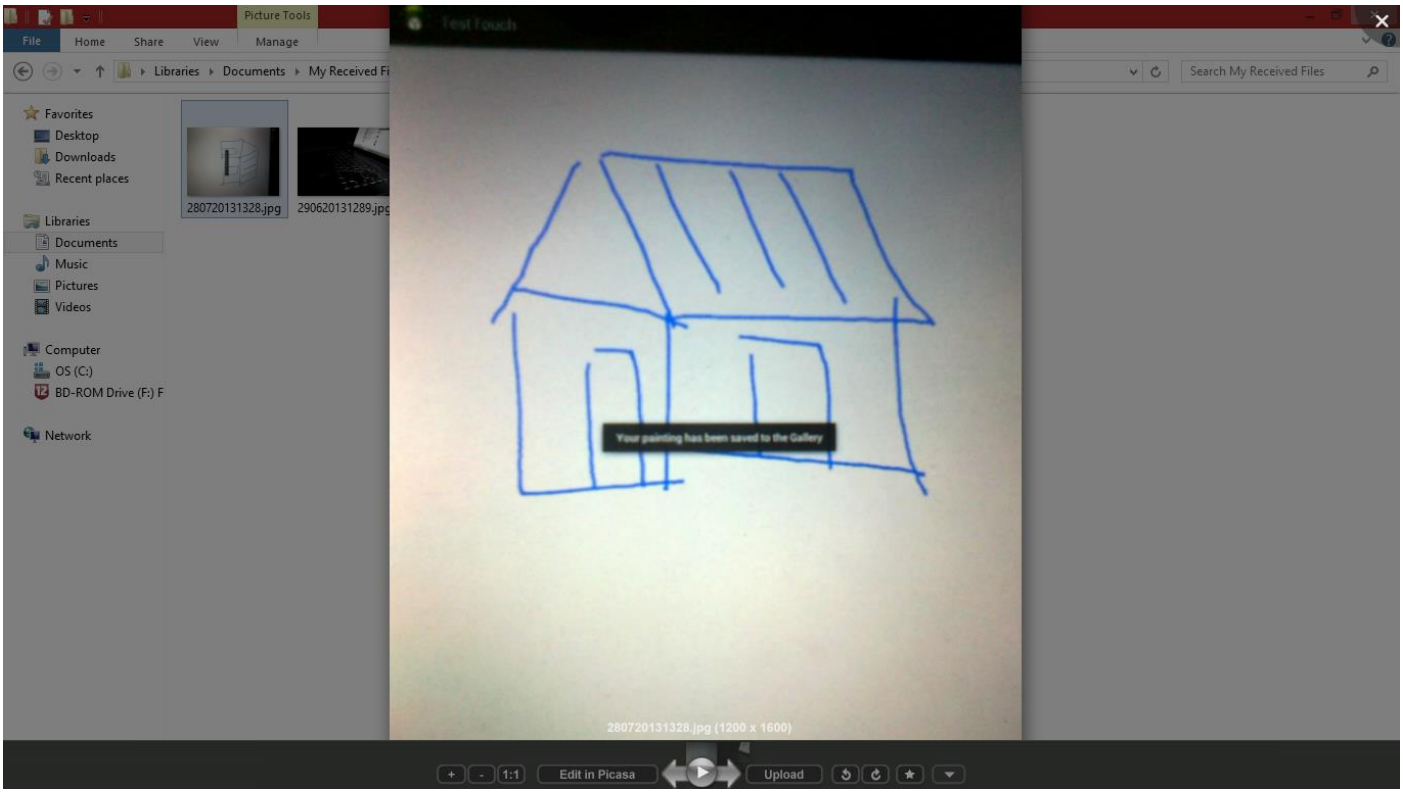
Screen Design 5: On selecting from Menu Button Color and save option are highlighted



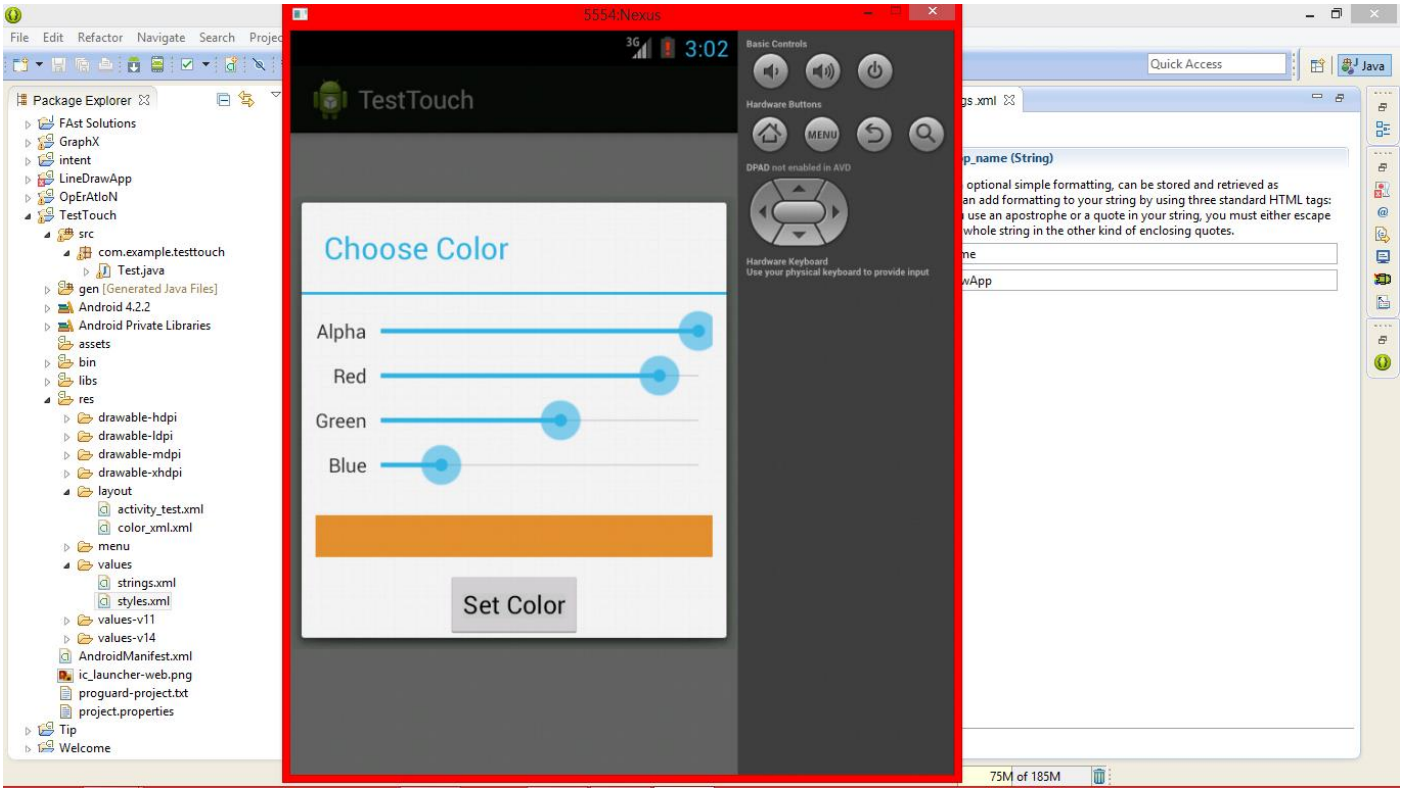
Screen Design 6: A drawing of a house with the mouse



Screen Design 7: Saving of an image at a predefined loacation



Screen Design 8: Color changing options



activity-main.xml(front page)

Name(of components)	Forecolor	Backcolor	Height	Width	Action
IMAGE VIEW		White	Match Parent	Match Parent	Draw

CODING

```
package com.example.testtouch;

import java.io.IOException;
import java.io.OutputStream;
import java.util.concurrent.atomic.AtomicBoolean;

import android.view.MenuInflater;
import android.app.Activity;
import android.app.Dialog;
import android.content.ContentValues;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore.Images;
import android.view.Display;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.Button;
```

```
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.Toast;

public class Test extends Activity implements OnTouchListener,
        SensorEventListener {
    ImageView imageView;
    Bitmap bitmap;
    Canvas canvas;
    Paint paint;
    Path path;
    float downx = 0, downy = 0, upx = 0, upy = 0;
    private float lastAcceleration;
    private float currentAcceleration;
    private float acceleration;
    private float movex;
    private float movey;
    private float oldmovey;
    private float oldmovex;
    private Dialog currentDialog;
    private AtomicBoolean dialogIsVisible = new AtomicBoolean(); // false

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_test);
        path = new Path();
        imageView = (ImageView) this.findViewById(R.id.ImageView);
        acceleration = 0.0f;
        currentAcceleration = SensorManager.GRAVITY_EARTH;
        lastAcceleration = SensorManager.GRAVITY_EARTH;
        Display currentDisplay = getWindowManager().getDefaultDisplay();
        float dw = currentDisplay.getWidth();
        float dh = currentDisplay.getHeight();
        SensorManager sensor = (SensorManager) getSystemService(SENSOR_SERVICE);
        sensor.registerListener(this,
                sensor.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_NORMAL);

        bitmap = Bitmap.createBitmap((int) dw, (int) dh,
                Bitmap.Config.ARGB_8888);
        canvas = new Canvas(bitmap);
        paint = new Paint();
        paint.setAntiAlias(true);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(4);
        paint.setStrokeCap(Paint.Cap.SQUARE);
        paint.setColor(Color.BLUE);

        imageView.setImageBitmap(bitmap);

        imageView.setOnTouchListener(this);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.activity_test, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
        switch (item.getItemId()) {
        case R.id.item1:
                // create the dialog and inflate its content
                currentDialog = new Dialog(this);
                currentDialog.setContentView(R.layout.color_xml);
                currentDialog.setTitle(R.string.title_color_dialog);
                currentDialog.setCancelable(true);

                // get the color SeekBars and set their onChange listeners
                final SeekBar alphaSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.alphaSeekBar);
                final SeekBar redSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.redSeekBar);
                final SeekBar greenSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.greenSeekBar);
                final SeekBar blueSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.blueSeekBar);

                // register SeekBar event listeners
                alphaSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);
                redSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);
                greenSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);
                blueSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);

                // use current drawing color to set SeekBar values
                final int color = paint.getColor();
                alphaSeekBar.setProgress(Color.alpha(color));
                redSeekBar.setProgress(Color.red(color));
                greenSeekBar.setProgress(Color.green(color));
                blueSeekBar.setProgress(Color.blue(color));

                // set the Set Color Button's onClickListener
                Button setColorButton = (Button) currentDialog
                                .findViewById(R.id.setColorButton);
                setColorButton.setOnClickListener(setColorButtonListener);

                dialogIsVisible.set(true); // dialog is on the screen
                currentDialog.show(); // show the dialog
                return true;

                // More items go here (if any) ...

        case R.id.item2:
                saveImage();
                return true;
        }
        return false;
}

private OnClickListener setColorButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
                // get the color SeekBars
                SeekBar alphaSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.alphaSeekBar);
                SeekBar redSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.redSeekBar);
                SeekBar greenSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.greenSeekBar);
                SeekBar blueSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.blueSeekBar);

                // set the line color
                paint.setColor(Color.argb(alphaSeekBar.getProgress(),
                                redSeekBar.getProgress(), greenSeekBar.getProgress(),
```

```
                        blueSeekBar.getProgress()));
                dialogIsVisible.set(false); // dialog is not on the screen
                currentDialog.dismiss(); // hide the dialog
                currentDialog = null; // dialog no longer needed
        } // end method onClick
}; // end setColorButtonListener

private OnSeekBarChangeListener colorSeekBarChanged = new OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress,
                        boolean fromUser) {
                // get the SeekBars and the colorView LinearLayout
                SeekBar alphaSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.alphaSeekBar);
                SeekBar redSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.redSeekBar);
                SeekBar greenSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.greenSeekBar);
                SeekBar blueSeekBar = (SeekBar) currentDialog
                                .findViewById(R.id.blueSeekBar);
                View colorView = (View) currentDialog.findViewById(R.id.colorView);

                // display the current color
                colorView.setBackgroundColor(Color.argb(alphaSeekBar.getProgress(),
                                redSeekBar.getProgress(), greenSeekBar.getProgress(),
                                blueSeekBar.getProgress()));
        } // end method onProgressChanged

        // required method of interface OnSeekBarChangeListener
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        } // end method onStartTrackingTouch

        // required method of interface OnSeekBarChangeListener
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        } // end method onStopTrackingTouch
}; // end colorSeekBarChanged

public boolean onTouch(View v, MotionEvent event) {
        int action = event.getAction();
        switch (action) {
        case MotionEvent.ACTION_DOWN:
                downx = event.getX();
                downy = event.getY();
                oldmovex = downx;



                oldmovey = downy;
                path.moveTo(downx, downy);
                break;
        case MotionEvent.ACTION_MOVE:
                movex = event.getX();
                movey = event.getY();
                path.quadTo(oldmovex, oldmovey, movex, movey);
                oldmovex = movex;
                oldmovey = movey;
                canvas.drawPath(path, paint);
                imageView.invalidate();
                path.reset();
                path.moveTo(movex, movey);

                break;
        case MotionEvent.ACTION_UP:
```

```
                upx = event.getX();
                upy = event.getY();
                path.moveTo(upx, upy);
                break;
        case MotionEvent.ACTION_CANCEL:
                break;
        default:
                break;
        }

        return true;
}

@Override
public void onAccuracyChanged(Sensor arg0, int arg1) {
        // TODO Auto-generated method stub

}

@Override
public void onSensorChanged(SensorEvent arg0) {
        // TODO Auto-generated method stub
        float x = arg0.values[0];
        float y = arg0.values[1];
        float z = arg0.values[2];

        lastAcceleration = currentAcceleration;
        currentAcceleration = x * x + y * y + z * z;

        acceleration = currentAcceleration
                        * (currentAcceleration - lastAcceleration);
        if (acceleration > 200000) {
                bitmap.eraseColor(Color.WHITE);
                imageView.invalidate();
        }
}

// save the current image to the Gallery
public void saveImage() {

        String fileName = "TT" + System.currentTimeMillis();

        // create a ContentValues and configure new image's data
        ContentValues values = new ContentValues();
        values.put(Images.Media.TITLE, fileName);
        values.put(Images.Media.DATE_ADDED, System.currentTimeMillis());
        values.put(Images.Media.MIME_TYPE, "image/jpg");

        // get a Uri for the location to save the file
        Uri uri = getContentResolver().insert(
                        Images.Media.EXTERNAL_CONTENT_URI, values);

        try {
                // get an OutputStream to uri
                OutputStream outStream = getContentResolver()
                                .openOutputStream(uri);

                // copy the bitmap to the OutputStream
                bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outStream);

                // flush and close the OutputStream
                outStream.flush(); // empty the buffer
                outStream.close(); // close the stream

                // display a message indicating that the image was saved
```

```
                    Toast message = Toast.makeText(this,
                                R.string.message_saved, Toast.LENGTH_SHORT);
                    message.setGravity(Gravity.CENTER, message.getXOffset() / 2,
                                message.getYOffset() / 2);
                    message.show(); // display the Toast
             } // end try
             catch (IOException ex) {
                    // display a message indicating that the image was saved
                    Toast message = Toast.makeText(this,
                                R.string.message_error_saving, Toast.LENGTH_SHORT);
                    message.setGravity(Gravity.CENTER, message.getXOffset() / 2,
                                message.getYOffset() / 2);
                    message.show(); // display the Toast
             } // end catch
      } // end method saveImage
}
```

CODING STANDARDS FOLLOWED AND ASSUMPTIONS

Coding standards should be developed depending on what suits best for the team and the specific types of products they develop. From the list of some general coding standards and guidelines we can get an idea about the types of coding standards that are being used.

Some General Coding Standards:

- **Declaration of global data** - What types of data can and cannot be declared as global are listed with a view to limit the data that needs to be defined with global scope.

- **Standard headers for different modules of the code** - The exact format in which the header information is organized in the header is specified. The information which is stored in the headers of different modules should be standard for an organization.

- **Naming conventions for variables** - A standard naming convention should be followed for naming all the global and local variables and constant identifiers.

- **Conventions regarding error return values and exception handling mechanisms** - The way error conditions are reported by different functions in a program should be standardized.

Coding Guidelines:

- Code should be easy to understand. As a result maintenance and debugging will be easier.

- The side effects of a function call include modifications to the parameters passed by reference, modification of global variables and I/O operations. Such side effects should be avoided as they make it difficult to understand a piece of code.

- Uses of an identifier for multiple purposes should be avoided.

- Uses of comment lines in the code will help in documenting the code properly. As a result a well-documented code makes it easier to understand.

- Lengthy functions in the code make it difficult to understand.

- Use of GO TO statements makes a program unstructured, thereby making it difficult to understand, debug and maintain the program.

TESTING

- ## Unit Testing

Unit testing is undertaken when a module has been coded and successfully reviewed. This can be done by two methods:

a) Black Box testing

b) Equivalence Class Partitioning

a) Black Box Testing

Test cases are designed from an examination of the input/output values only and no knowledge of designing or coding is required the following are the two main approaches of designing black-box test cases.

b) Equivalence Class Partitioning

The domain of input values to a program is partitioned into a set of equivalence classes. This partitioning is done in such a way that the behavior of the program is similar to every boundary value analysis. Boundary value analysis leads to selection of the test cases at the boundaries of different equivalence classes.

- ## White Box Testing

There are several white box testing methods as given below:

a) Statement coverage

This strategy aims to design test cases such that every statement in a program is executed at least once.

b) Branch coverage

Test cases are designed to make each branch condition assume both true and false value, such that each branch is executed once.

c) Condition coverage

Test cases are designed to make each component of a composite conditional expression assume both true and false values.

d) Path coverage

Path coverage testing strategy requires us to design such that all linearly independent paths of a program are executed at least once.

- **Integration Testing :**

The objective of integration testing is to check whether the different modules of a program interface with each other properly. Suring this testing, different modules of a system are integrated in a planned manner. After each integration step, the partially integrated system is tested.

In this project the different modules are tested when they are available to realize the full system. Thereby the mixed approach of integration testing is followed.

- **System Testing :**

After integration testing, the system testing is carried out on the fully integrated system. It is done to validate a fully developed system to assure that it meets its requirements as stated in the SRS document.

Here alpha testing is carried out on the project. It is a type of system testing which is carried out by the test team within the developing organization.

So the project developers carry out the testing themselves to ensure that the requirements have been successfully met.

FUTURE SCOPE

Android Apps are fairly complex to develop due to fragmentation, with manufacturers running a different version of Android. Also with the existence of multiple customized user interfaces and hardware, Android App developers have a hard time developing and maintaining code for their apps. This application is merely developed for all android based devices that can be either A gingerbread, ice-cream sandwich & jelly bean and etc. This application can be redeveloped or a module can be added so that drawing a picture with filling colors from pallets. Further it can be made much more adding brushing color and applying a background color.

REFERENCES AND BIBLIOGRAPHY

1. *Ardent Community – www.ardentcollaborations.com.community*
2. developer.**android**.com
3. *www.vogella.com/articles/**Android**/*
4. *www.mkyong.com/**tutorials**/**android**-**tutorial**/*
5. *www.**tutorials**point.com/**android**/*
6. *www.coreservlets.com/**android**-**tutorial**/*
7. *Book – Hello Android by Ed Burnette*
8. *Java Book by Herbart Schildt*
9. *Java Book by Kathy Sierra*