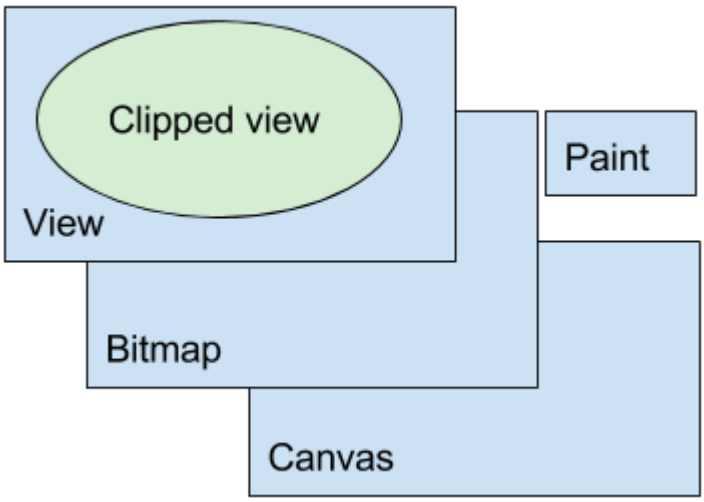
Canvas 2D

In a simple layman language, a canvas is like a blank place like a blank paper /material where someone can write/paint on.

A Canvas works for you as a pretence, or interface, to the actual surface upon which your graphics will be drawn — it holds all your "draw" calls. Via the Canvas, your drawing is actually performed upon an underlying Bitmap, which is placed into the window. To draw something, you need 4 basic components:

- A canvas object. Very simplified, a Canvas is a logical 2D drawing surface that provides methods for drawing onto a bitmap.
- An instance of the Bitmap class which represents the physical drawing surface and gets pushed to the display by the GPU.
- A View instance associated with the bitmap.
- A Paint object that holds the style and colour information about how to draw geometries, text, and on bitmap.

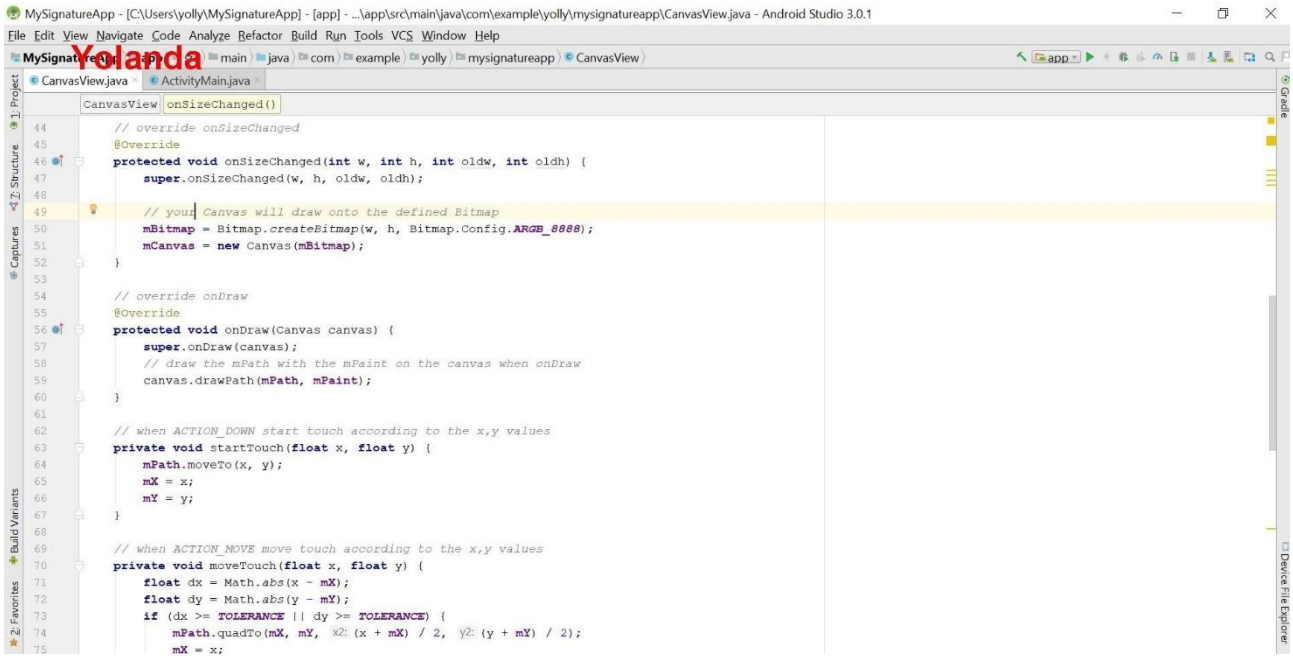


You can associate a Canvas with an ImageView and draw on it in response to user actions. This basic implementation of drawing does not require a custom View. You create an app with a layout that includes an ImageView that has a click handler. You implement the click handler in MainActivity to draw on and display the Canvas.

Drawing to a Canvas, is better when your application needs to regularly re-draw itself.
The Canvas class has its own set of drawing methods that you can use,

like drawBitmap(...), drawPath(...), drawText(...), and many more. Other classes that you might use also have draw()methods.

For example, you'll probably have some Drawable objects that you want to put on the Canvas. Drawable has its own draw() method that takes your Canvas as an argument.



The code above in detail.

In this code snippet, we set up a new Canvas. This canvas will draw onto the defined Bitmap.

```
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        mBitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
        mCanvas = new Canvas(mBitmap);
}
```

Now that we have the canvas we get the x and y event coordinates to make our path moves.

```
@Override
public boolean onTouchEvent(MotionEvent event) {
        float x = event.getX();
        float y = event.getY();

        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
                startTouch(x, y);
                invalidate();
                break;
        case MotionEvent.ACTION_MOVE:
                moveTouch(x, y);
```

```
                invalidate();
                break;
        case MotionEvent.ACTION_UP:
                upTouch();
                invalidate();
                break;
        }
        return true;
}
```

We transform the x, y event coordinates into path moves.

```
private void moveTouch(float x, float y) {
        float dx = Math.abs(x - mX);
        float dy = Math.abs(y - mY);
        if (dx >= TOLERANCE || dy >= TOLERANCE) {
                mPath.quadTo(mX, mY, (x + mX) / 2, (y + mY) / 2);
                mX = x;
                mY = y;
        }
}
```

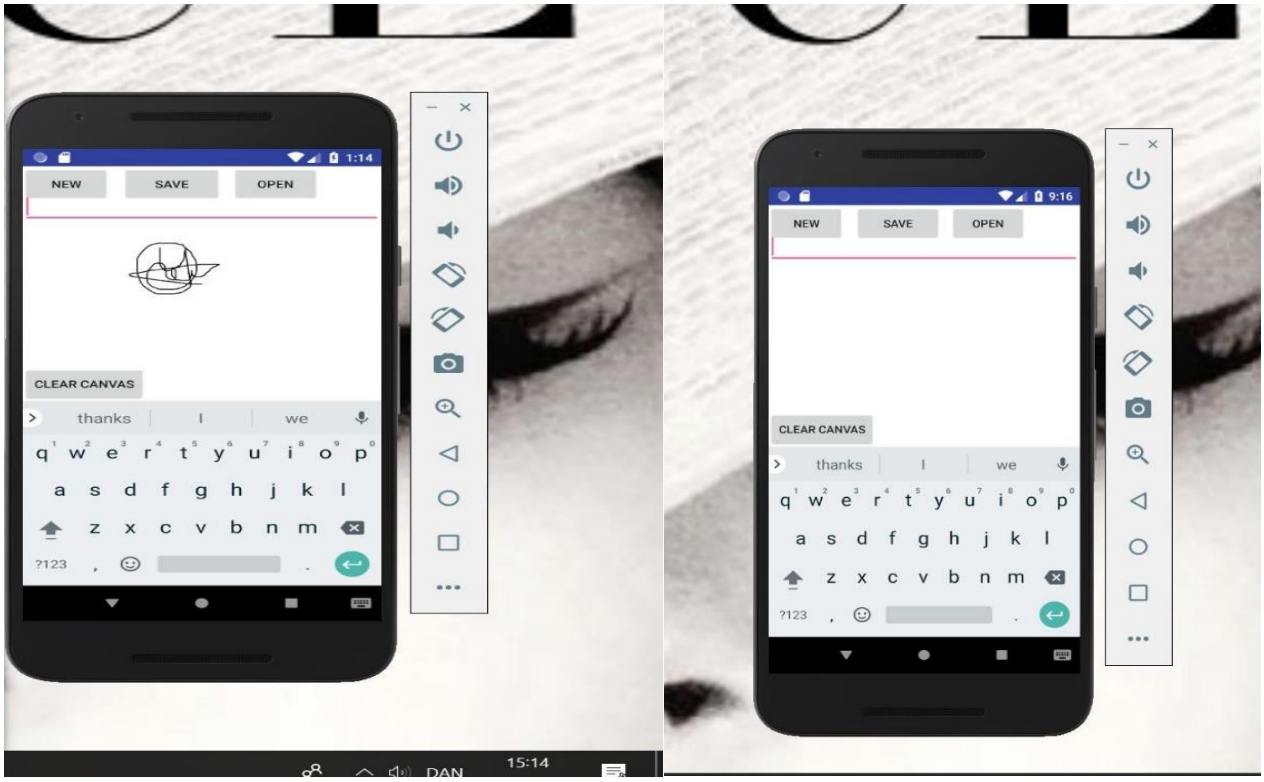
And by overriding the onDraw event, we draw our path onto the canvas. By establishing the path your finger is drawing and painting on it.

```
// override onDraw
@Override
protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawPath(mPath, mPaint);
}
```

The clear Canvas Method is called to clear the drawing, in it we call an invalidate method which forces re-draw.

In summary

You are creating a view class then extends View. You override the onDraw(). You add the path of where finger touches and moves. You override the onTouch() of this purpose. In your onDraw() you draw the paths using the paint of your choice. You should call invalidate() to refresh the view.



With the knowledge I have acquired on canvas I intended to use canvas in creating a 2D graphics app. I managed to create a signature app where someone can write his/her signature save it and paste it on a document. I decided on the signature app because I have had certain instance where I had to print a document and put my signature then scan the document and send, and at that moment my printer was not working so it took time for me to print sign and scan and send the document. With the app it is a matter of seconds no printer or scanner is needed so it saves time and money.

Though it is not a complete project, but I managed to create the signature app where someone can sign and clear screen just in case one makes a mistake