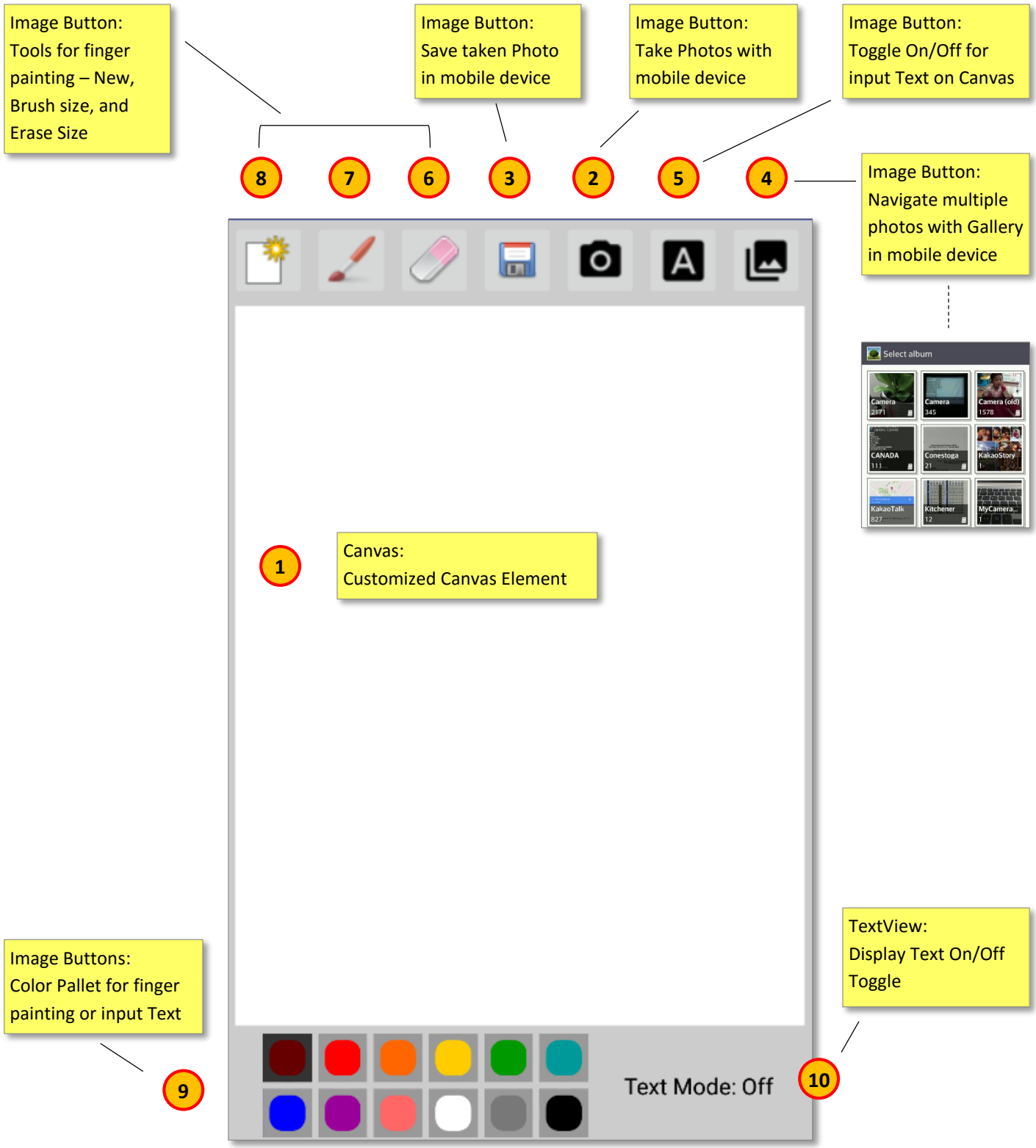
INFO8250 - Mobile Application Design

Assignment 2

Image Processing

SungJoe KIM (7120082)

Assignment 2 - User Interface (Image Processing)



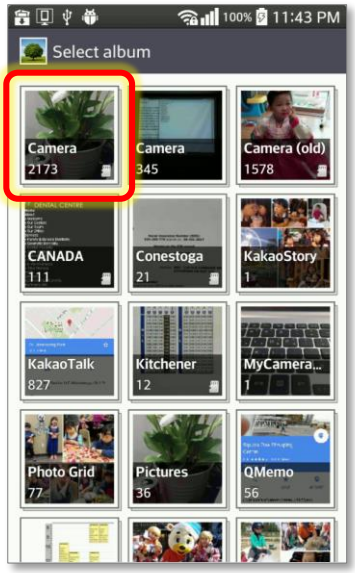
A paragraph and supporting evidence for each functional requirement

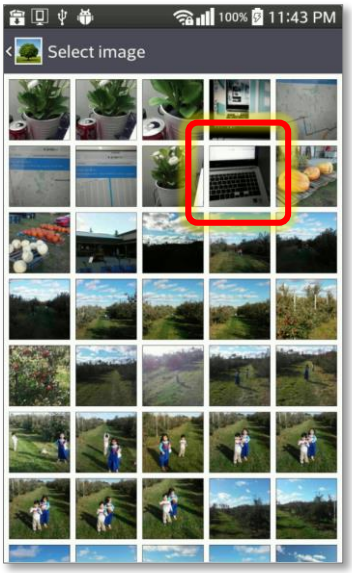
In order to meet with requirements, I created a customized Canvas element as a class named DrawingView.java.

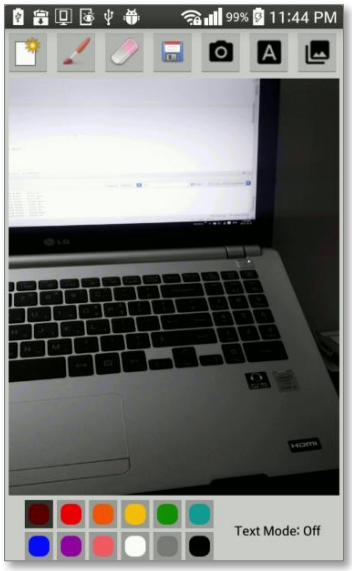
1. Take photos with a mobile device

Using camera application, Intent object, and startActivityForResult() method, code to handle the bitmap image data and save the full-size image to the internal storage of mobile device. After stored the image, open the photo image from local storage and draw on the customized Canvas.

Along with loading photo image just taken, Navigate the photos saved in the local mobile device and load on the Canvas.







The main code to implement these functional requirements is shown below:

```
if(view.getId()==R.id.camera_btn) {

    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            // Error occurred while creating the File
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
                    Uri.fromFile(photoFile));
            startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
        }
    }
```

```
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);

            // Get the Image from data
            Uri selectedImage = data.getData();
            String[] filePathColumn = { MediaStore.Images.Media.DATA };

            // Get the cursor
            Cursor cursor = getContentResolver().query(selectedImage,
                    filePathColumn, null, null, null);
            // Move to first row
            cursor.moveToFirst();

            int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
            imgDecodableString = cursor.getString(columnIndex);
            cursor.close();
            mCurrentPhotoPath = imgDecodableString;
            galleryAddPic();
            setPic();

}
```

```
private void galleryAddPic() {
    //String path = Environment.getExternalStorageDirectory().getPath();
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    File f = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    this.sendBroadcast(mediaScanIntent);
}
```

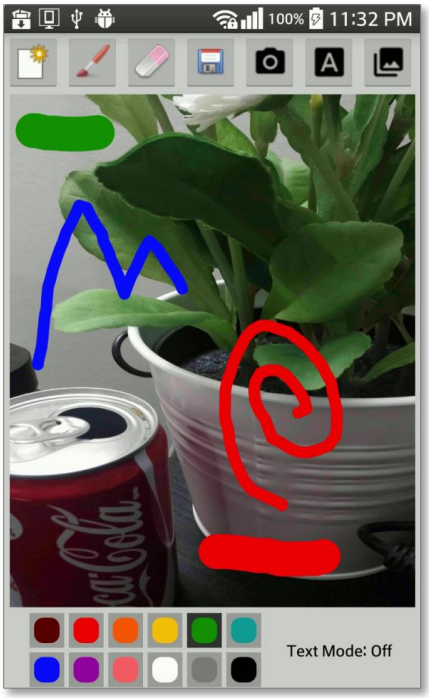
```
private void setPic() {

    DrawingView mDrawingView =(DrawingView)findViewById(R.id.drawing);

    // Get the dimensions of the View
    int targetW = 0;
    int targetH = 0;

    targetW = mDrawingView.getWidth();
    targetH = mDrawingView.getHeight();

    // Get the dimensions of the bitmap
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;

    BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;

    // Determine how much to scale down the image
    int scaleFactor = Math.min(photoW/targetW, photoH/targetH);

    // Decode the image file into a Bitmap sized to fill the View
    bmOptions.inJustDecodeBounds = false;
    bmOptions.inSampleSize = scaleFactor;
    bmOptions.inPurgeable = true;

    Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    bitmap = Bitmap.createScaledBitmap(bitmap,targetW,targetH,true);

    mDrawingView.drawBmp(bitmap);
}
```

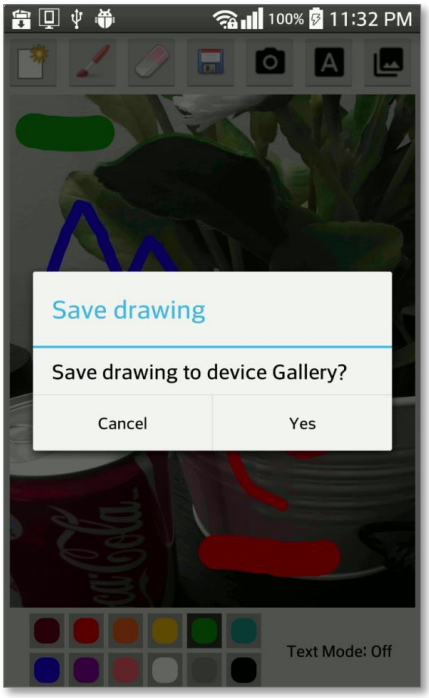
```
private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp =
         new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = "JPEG_" + timeStamp;
    File storageDir = Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES);
    File image = File.createTempFile(
            imageFileName,  /* prefix */
            ".jpg",         /* suffix */
            storageDir      /* directory */
    );
    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath =  image.getAbsolutePath();
    return image;
}
```

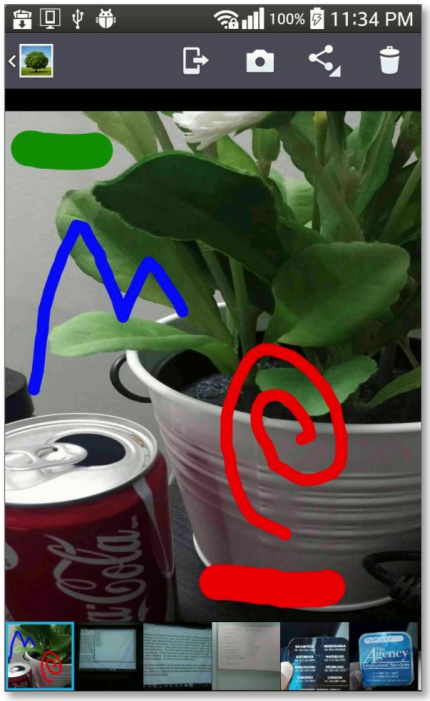
```
public void drawBmp(Bitmap bmp){

    int bw = bmp.getWidth();
    int bh = bmp.getHeight();

    drawCanvas.drawRect(0, 0, bw, bh, canvasPaint);
    drawCanvas.drawBitmap(bmp, null, new Rect(0, 0, bw, bh), null);
}
```

2. Sketch overlays on the photos

In order to draw on the white Canvas or loaded photo, implement override event methods such as onDraw(Canvas canvas), onTouchEvent(MotionEent event). When touch the 'Save' <ImageButton>, the image drawn on the canvas including sketch will be stored in the local storage, which can be found at the 'Gallery' application embedded in the mobile device. Along with finger painting, implemented additional functions such as brush color and size, and erase as well.







The main code to implement these functional requirements is shown below:

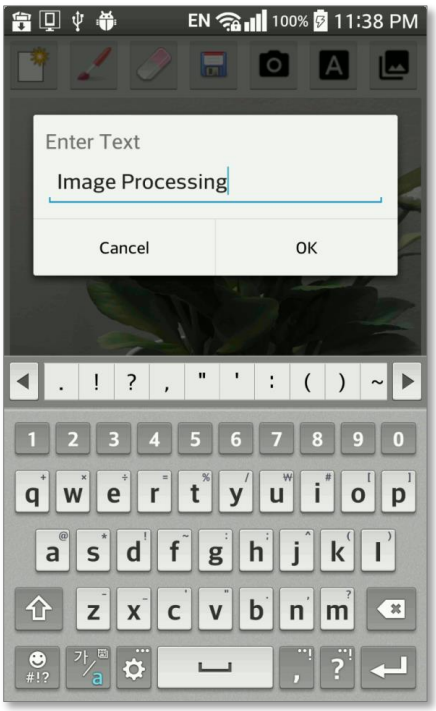
```
@Override
public boolean onTouchEvent(MotionEvent event) {
    //detect user touch
    float touchX = event.getX();
    float touchY = event.getY();

    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            drawPath.moveTo(touchX, touchY);
            break;
        case MotionEvent.ACTION_MOVE:
            drawPath.lineTo(touchX, touchY);
            break;
        case MotionEvent.ACTION_UP:
            drawCanvas.drawPath(drawPath, drawPaint);
            drawPath.reset();
            break;
        default:
            return false;
    }
    invalidate();
    return true;
}
```

```
public void paintClicked(View view){

    //use chosen color
    if(view!=currPaint){
        drawView.setErase(false);

        //update color
        ImageButton imgView = (ImageButton)view;
        if(!isTextMode){
            imgPaintView = imgView;
        }else{
            imgPaintText = imgView;
        }

        String color = view.getTag().toString();
        drawView.setColor(color);

imgView.setImageDrawable(getResources().getDrawable(R.drawable.paint_pressed));

currPaint.setImageDrawable(getResources().getDrawable(R.drawable.paint));
        currPaint=(ImageButton)view;
        drawView.setBrushSize(drawView.getLastBrushSize());
    }
}
```

```
public void setColor(String newColor){
    //set color
    invalidate();
    if(MainActivity.isTextMode){
        paintColor = Color.parseColor(newColor);
        textPaint.setColor(paintColor);
    }else{
        paintColor = Color.parseColor(newColor);
        drawPaint.setColor(paintColor);
        paintTempColor = paintColor;
    }
}

public void setBrushSize(float newSize){
    //update size
    float pixelAmount = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,
            newSize, getResources().getDisplayMetrics());
    brushSize=pixelAmount;
    drawPaint.setStrokeWidth(brushSize);
}
```

3. Add textual comments to the photos

In order to draw textual input value by user, added <ImageButton> for toggle of 'Text Mode' that allows for user to input text on the canvas or overlaid photo. During 'Text Mode On', touching on the canvas invokes onTouchEvent() which shows a AlertDialog element including <EditText> for input text by user. After text and click the 'OK' button in the Dialog, text input value will be draw at the position of touch. User also can change the color of text and save the image in the local storage.



The main code to implement these functional requirements is shown below:

```
@Override
public boolean onTouchEvent(MotionEvent event) {

    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            if(MainActivity.isTextMode) {
                positionX = (int) touchX;
                positionY = (int) touchY;
                mActivity.showInputDialog();
            }
            break;
        case MotionEvent.ACTION_MOVE:

          ...
    }

    invalidate();
    return true;
}
```

```
public void showInputDialog() {

    LayoutInflater layoutInflater = LayoutInflater.from(MainActivity.this);
    View promptView = layoutInflater.inflate(R.layout.input_dialog, null);
    AlertDialog.Builder alertDialogBuilder =
         new AlertDialog.Builder(MainActivity.this);
    alertDialogBuilder.setView(promptView);

    final EditText editText =
         (EditText) promptView.findViewById(R.id.edittext);

    // setup a dialog window
    alertDialogBuilder.setCancelable(false)
            .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {

                    drawView.drawText(editText.getText().toString());

                }
            })
            .setNegativeButton("Cancel",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            dialog.cancel();
                        }
                    });

    // create an alert dialog
    AlertDialog alert = alertDialogBuilder.create();
    alert.show();
}
```

```
public void drawText(String inputText){
    if(MainActivity.isTextMode){

        if(inputText !="" && inputText != null){

            drawCanvas.drawText(inputText, positionX, positionY, textPaint);
            inputText = "";
            positionX = 0;
            positionY = 0;
            invalidate();
        }
    }
}
```

AndroidMainifest.xml

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"  />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Development Environment: Android Studio

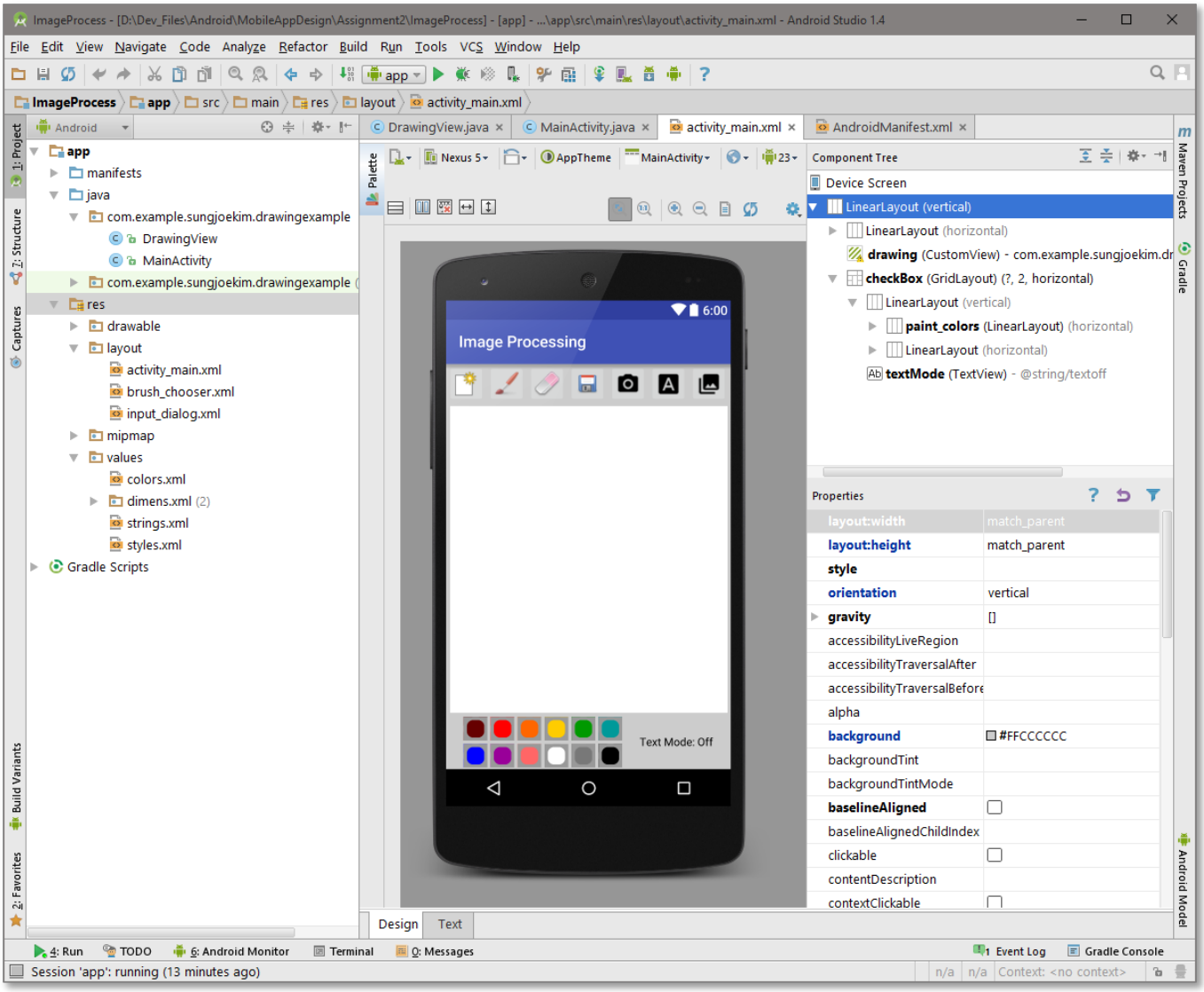


Image Processing