

# Mobile Image Processing

- Examples of mobile image processing
- Android platform
- Class resources for Android
- Eclipse integrated development environment
- Augmentation of viewfinder frames
- Debugging with DDMS perspective
- Taking a device screenshot
- Creating an Android emulator
- Mixed programming with Android JNI
- OpenCV library for Android
- Past class projects

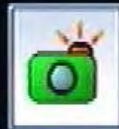
# Mobile landmark recognition



# Mobile product recognition

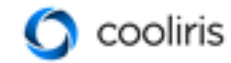


Query sent. Waiting for response.



# Android platform

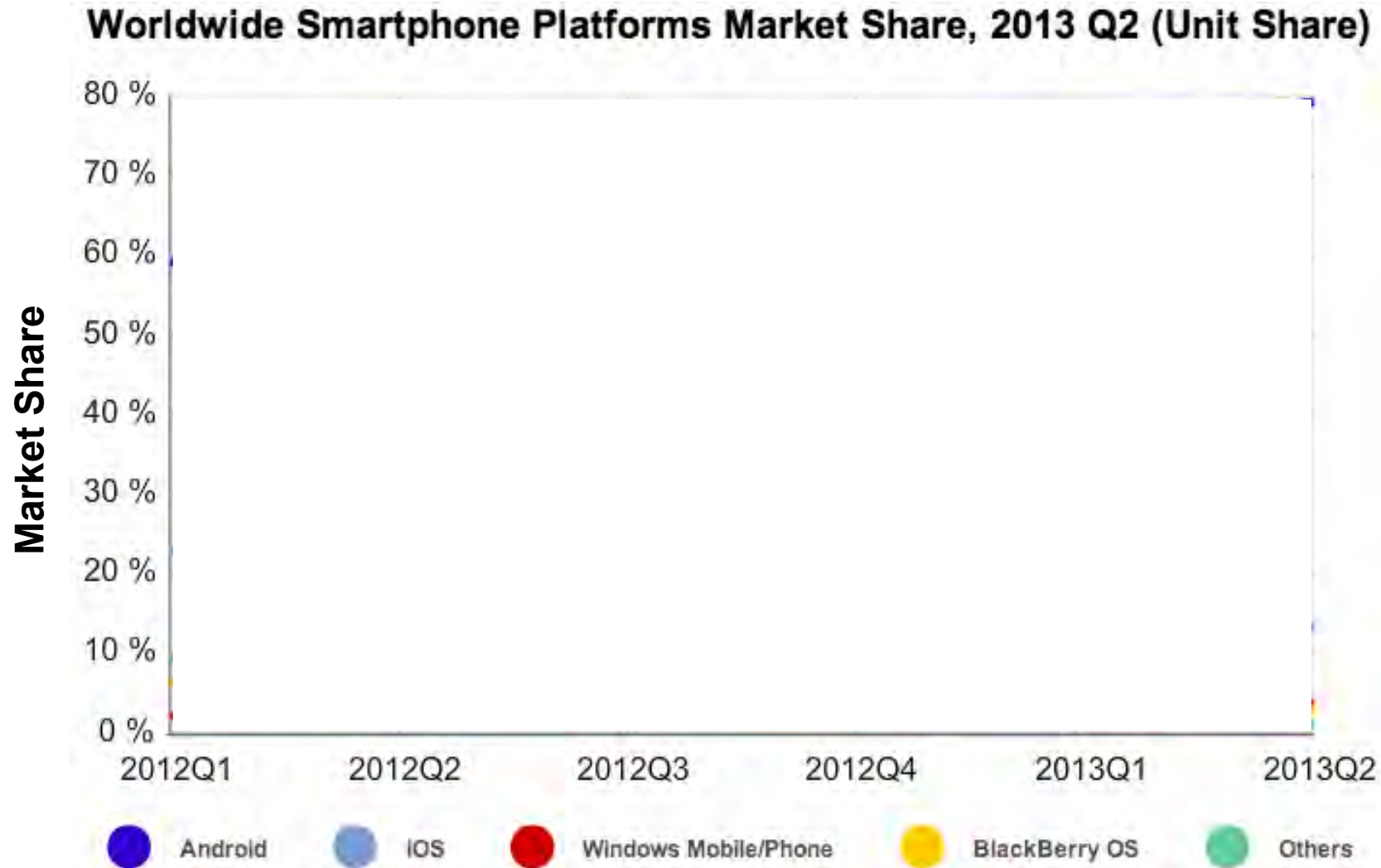
- Open source mobile platform developed by Google
- Supported and maintained by Open Handset Alliance
  - 14 mobile operators
  - 23 handset manufacturers
  - 21 semiconductor companies
  - 17 software makers



# Android platform

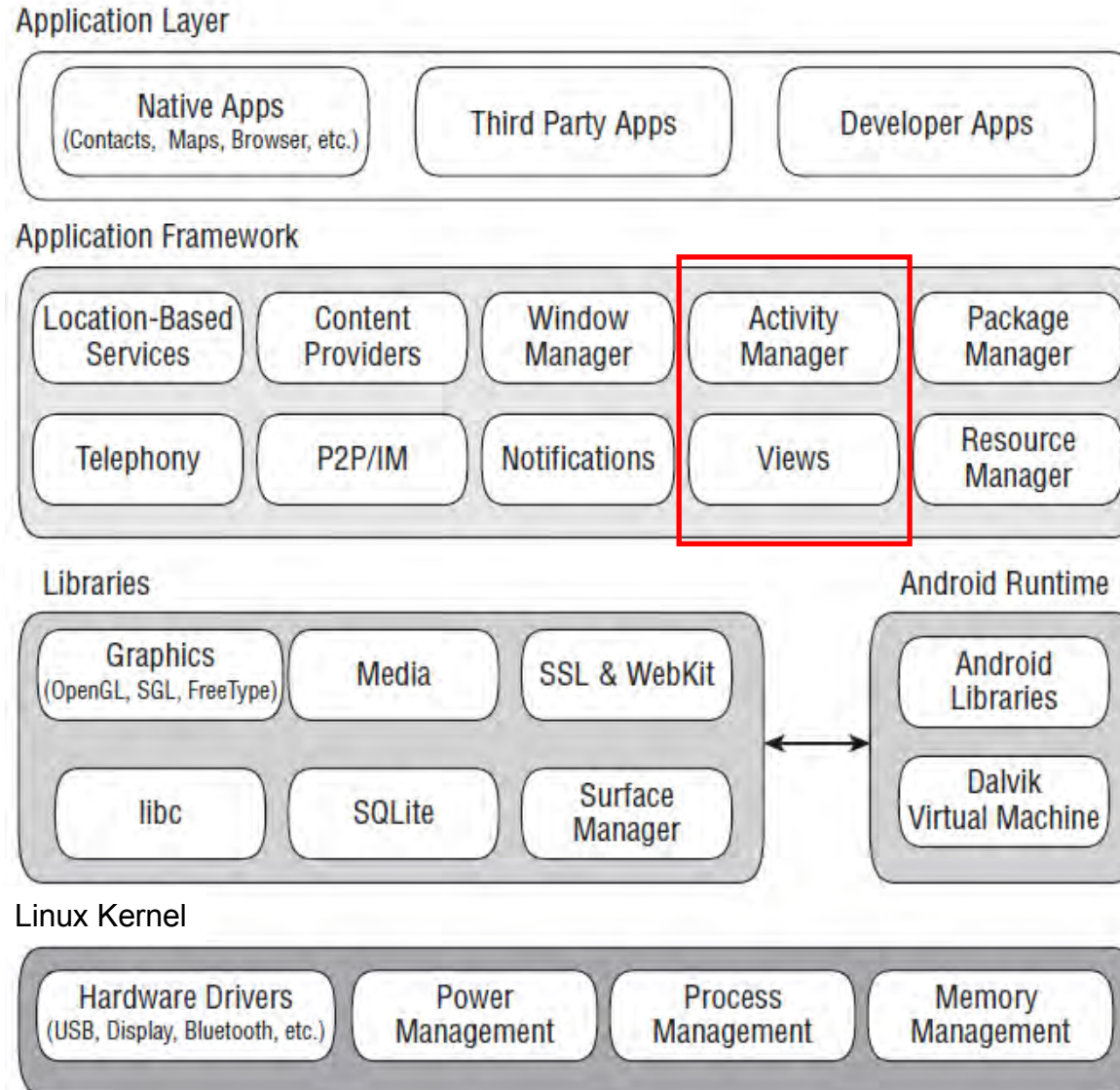
- Open source mobile platform developed by Google
- Supported and maintained by Open Handset Alliance
  - 14 mobile operators
  - 23 handset manufacturers
  - 21 semiconductor companies
  - 17 software makers
- Uses an open-source kernel and a virtual machine designed for mobile hardware
- Commands the largest share of the smartphone market in the world
- Google Play store contains over 1M apps

# Distribution of worldwide smartphone sales



*IDC Worldwide Mobile Phone Tracker*

# Android software stack



*Professional Android Application Development*

# Programming Android applications in Java

- Android encourages high-level app development
- Android uses Java as the main programming language
- Android inherits basic classes from conventional Java
  - String, Container, Math, IO, Network, ...
- Android also adds new classes specific to mobile devices
  - Camera, Telephony, Map, GPS, Speech, ...

# Class resources for mobile image processing

## ■ Tutorial #1: Basic Android Setup

- Image processing-oriented introduction to Android
- Explains how to download and install the different software packages (JDK, SDK, Eclipse) on your own computer or use these pre-installed packages on SCIEN machines
- Shows how to build and run viewfinder augmentation apps in real time



<http://www.stanford.edu/ee368/Android>

# Class resources for mobile image processing

- Tutorial #2: OpenCV for Android Setup
  - Builds on core skills developed in Tutorial #1
  - Explains how to download and install OpenCV for Android
  - Shows how to build viewfinder apps that detect circles and lines, detect feature keypoints, track feature keypoints, perform locally adaptive thresholding, detect human faces

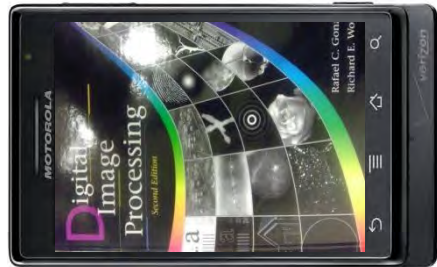


<http://www.stanford.edu/ee368/Android>

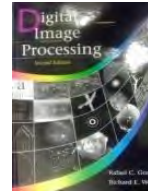
# Class resources for mobile image processing

- Tutorial #3: Server-client communications

1. Client takes an input image



2. Send data to server



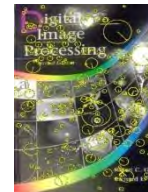
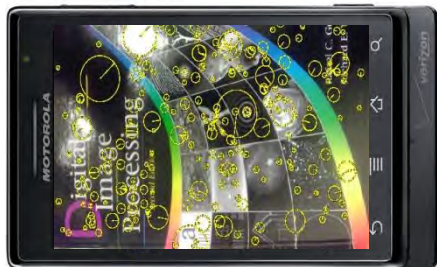
HTTP

3. A PHP server invokes application



4. Server-side application processes the data

5. Client receives the processed result



<http://www.stanford.edu/ee368/Android>

# Sample mobile image processing projects

Histogram Equalization



[Project Files \(zip\)](#)

Color Histograms



[Project Files \(zip\)](#)

Feature Tracking



[Project Files \(zip\)](#)

Locally Adaptive Binarization



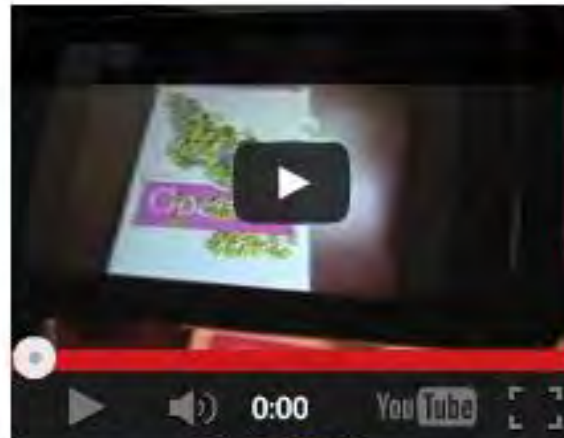
[Project Files \(zip\)](#)

Edges, Lines, and Circles



[Project Files \(zip\)](#)

Local Feature Keypoints



[Project Files \(zip\)](#)

Human Face Detection



[Project Files \(zip\)](#)

<http://www.stanford.edu/ee368/Android>

# Android development with Eclipse



# Eclipse integrated development environment

The screenshot shows the Eclipse IDE interface with several callout boxes pointing to specific features:

- Project files:** Points to the Package Explorer on the left, which displays a tree view of project files and folders.
- Text editor:** Points to the central workspace where Java source code is being edited.
- Different perspectives:** Points to the top toolbar, which contains icons for switching between different IDE perspectives.
- Class hierarchy:** Points to the Outline view on the right, which shows the class hierarchy and member lists for the current project.
- Errors and warnings:** Points to the Problems view at the bottom, which lists any compilation errors or warnings.

```
Paint mPaint;
Paint mPaintYellow;
byte[] mYUVData;
int[] mRGBData;
int mImageWidth, mImageHeight;
int[] mGrayHistogram;
double[] mGrayCDF;
int mState;

static final int STATE_ORIGINAL = 0;
static final int STATE_PROCESSED = 1;

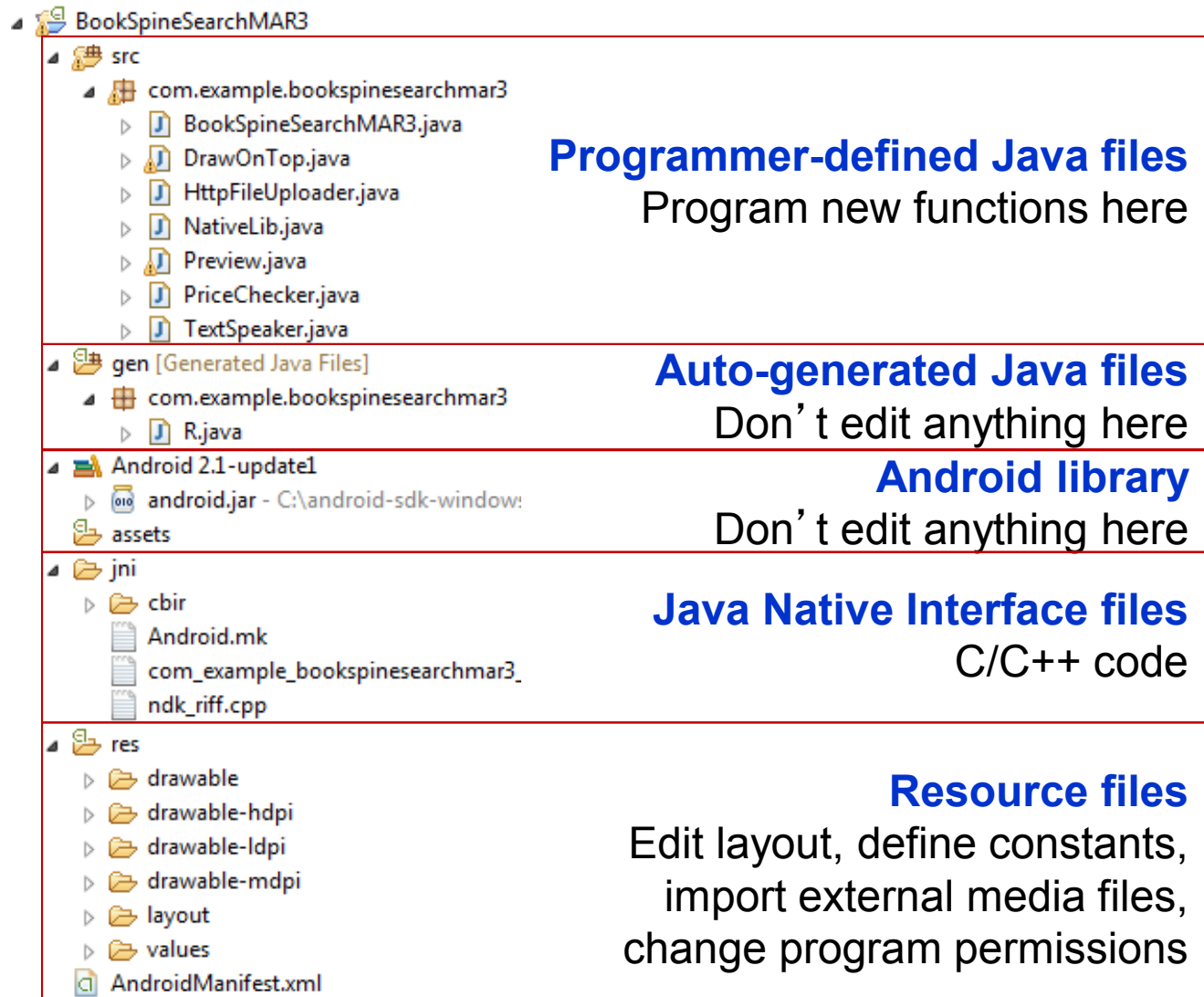
public DrawOnTop(Context context) {
    super(context);

    mPaintBlack = new Paint();
    mPaintBlack.setStyle(Paint.Style.FILL);
    mPaintBlack.setColor(Color.BLACK);
    mPaintBlack.setTextSize(25);

    mPaintYellow = new Paint();
    mPaintYellow.setStyle(Paint.Style.FILL);
    mPaintYellow.setColor(Color.YELLOW);
    mPaintYellow.setTextSize(25);
}
```

Description	Resolution
Attribute minSdkVersion (6) is lower than the project target API level (8)	Andr
Attribute minSdkVersion (7) is lower than the project target API level (8)	Andr
Attribute minSdkVersion (7) is lower than the project target API level (8)	Andr
The import android.os.KeyEvent is never used	CVCe
The import android.view.KeyEvent is never used	CVCe
The value of the local variable mImageHeight is not used	Draw

# Structure of an Android project



The screenshot shows the project structure of an Android application named 'BookSpineSearchMAR3'. The structure is organized into several key folders and files, each with a specific purpose:

- src**: Contains programmer-defined Java files where new functions are implemented. Files include `com.example.bookspinesearchmar3` (package), `BookSpineSearchMAR3.java`, `DrawOnTop.java`, `HttpFileUploader.java`, `NativeLib.java`, `Preview.java`, `PriceChecker.java`, and `TextSpeaker.java`.
- gen [Generated Java Files]**: Contains auto-generated Java files that should not be edited. It includes `com.example.bookspinesearchmar3` (package) and `R.java`.
- Android 2.1-update1**: Contains the Android library files, including `android.jar` (located at `C:\android-sdk-window`) and the `assets` folder. These should not be edited.
- jni**: Contains Java Native Interface files for C/C++ code. It includes the `cbir` folder, `Android.mk`, `com_example_bookspinesearchmar3`, and `ndk_riff.cpp`.
- res**: Contains resource files for editing layouts, defining constants, importing external media, and changing permissions. It includes `drawable`, `drawable-hdpi`, `drawable-ldpi`, `drawable-mdpi`, `layout`, `values`, and `AndroidManifest.xml`.

**Programmer-defined Java files**  
Program new functions here

**Auto-generated Java files**  
Don't edit anything here

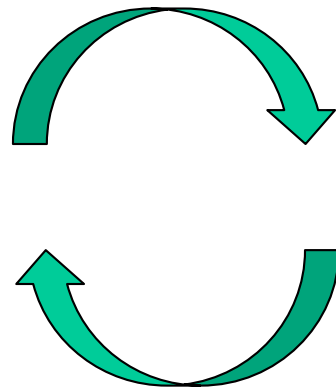
**Android library**  
Don't edit anything here

**Java Native Interface files**  
C/C++ code

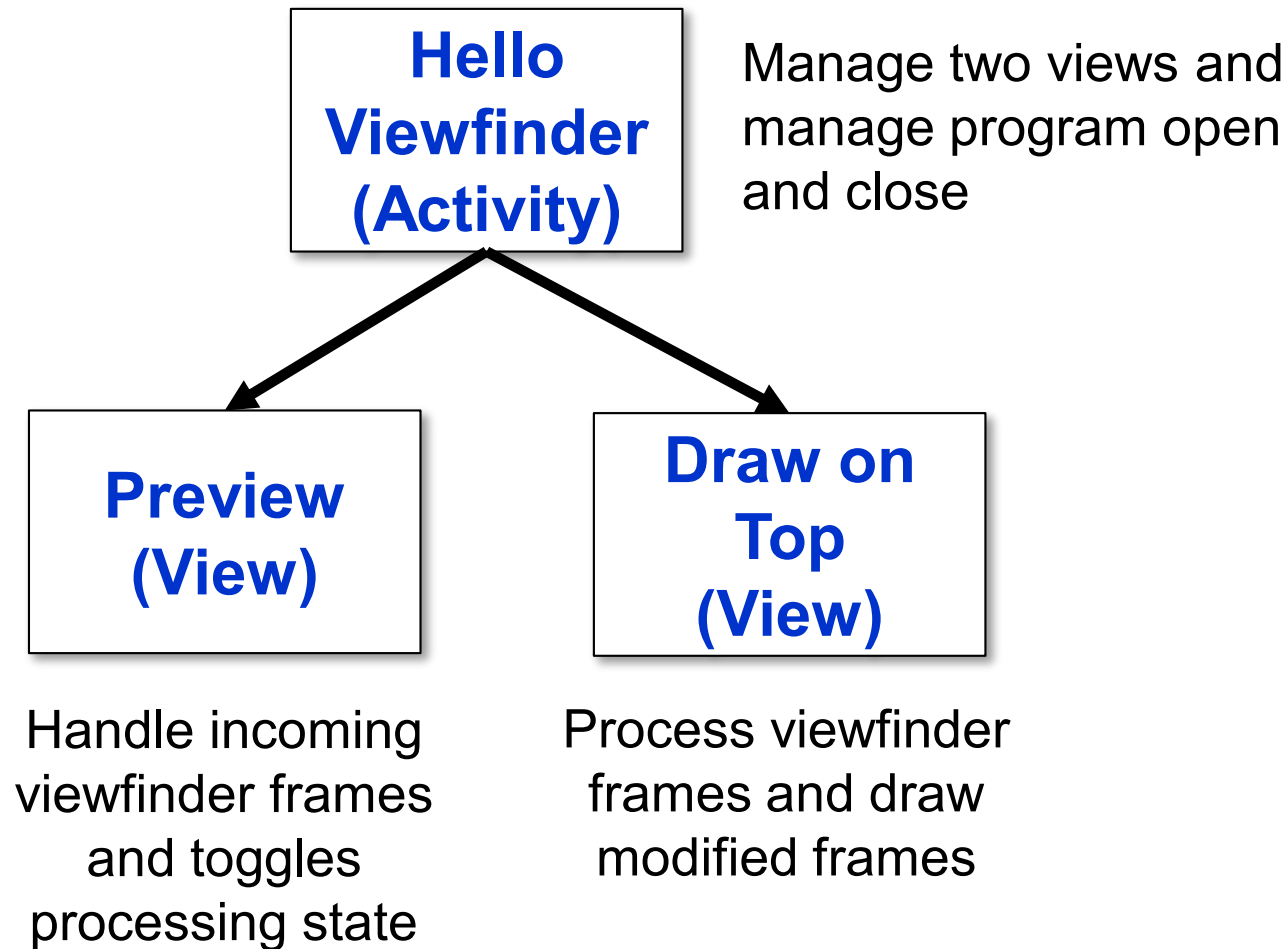
**Resource files**  
Edit layout, define constants,  
import external media files,  
change program permissions

# “Hello Viewfinder” project

- Goals of this project
  - Learn how to create a simple Android project
  - Learn how to display viewfinder frames
  - Learn how to process viewfinder frames
- Full source available on class website



# “Hello Viewfinder” class hierarchy



# Main activity class

```
public class HelloViewfinderActivity extends Activity {  
    private Preview mPreview;  
    private DrawOnTop mDrawOnTop;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // Hide the window title and set full screen  
        getWindow().setFlags(... Full Screen Parameters ...);  
        requestWindowFeature(Window.FEATURE_NO_TITLE);  
  
        // Create Preview and DrawOnTop  
        mDrawOnTop = new DrawOnTop(this);  
        mPreview = new Preview(this, mDrawOnTop);  
        setContentView(mPreview);  
        addContentView(mDrawOnTop, ... Layout Options ...)  
    }  
}
```

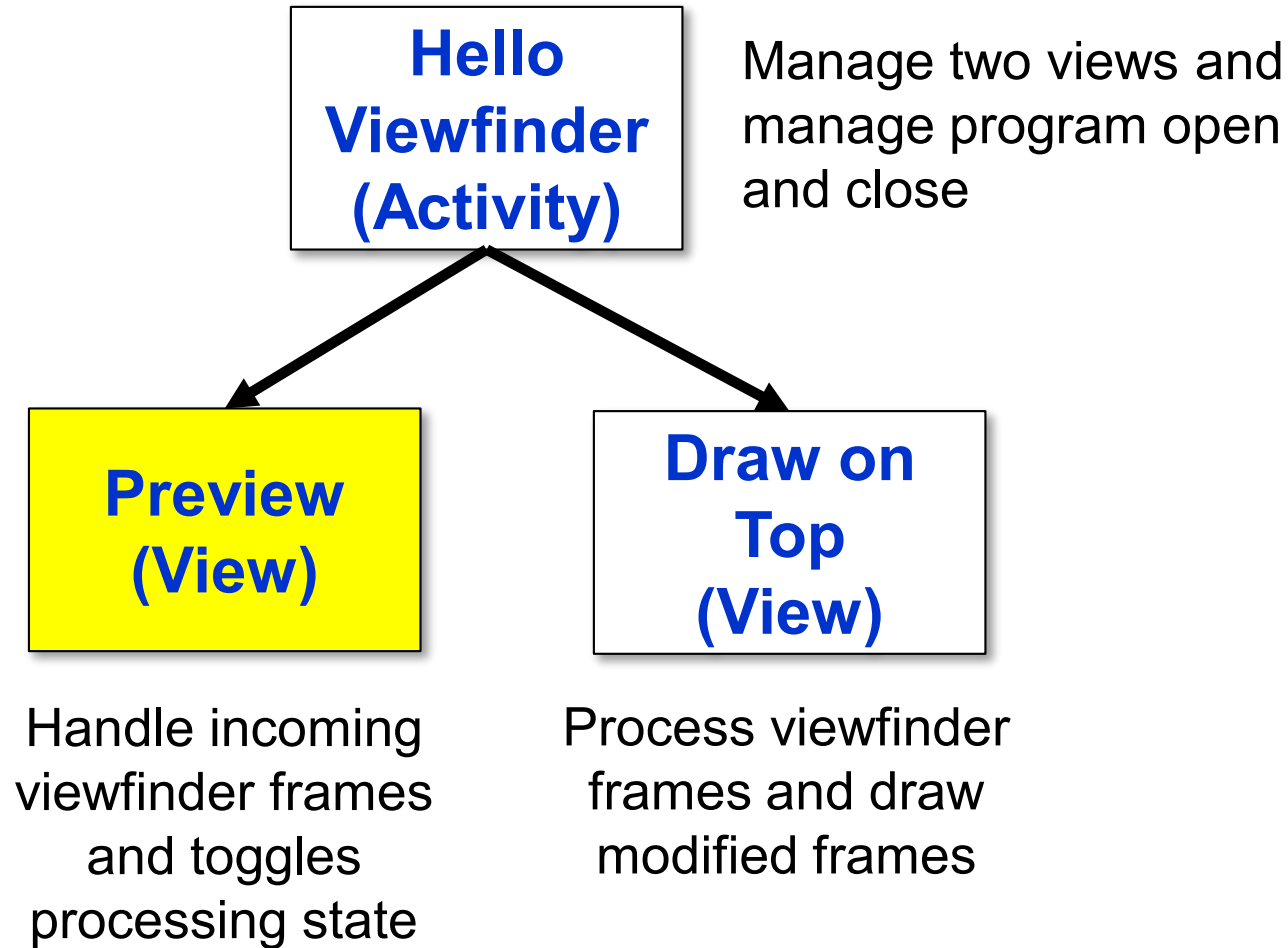
Make this class inherit the properties of an Activity

Called when the activity is first created

Make the application appear in full screen

Create two children objects for displaying frames

# “Hello Viewfinder” class hierarchy

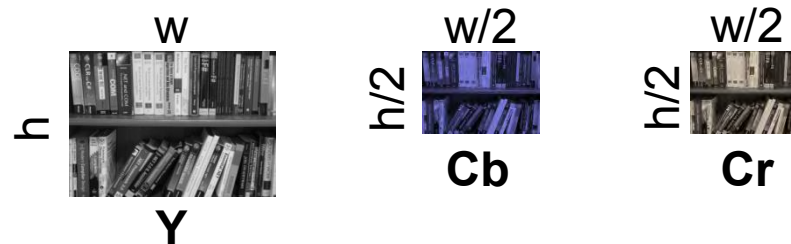


# Preview class: viewfinder frames go down two paths



```
myCamera.setPreviewCallback(new PreviewCallback() {  
    public void onPreviewFrame(byte[] data, Camera camera)  
    { ... Pass data to DrawOnTop class ... }  
});
```

## Data in YCbCr 4:2:0 format



# Preview class: toggle states via touch screen

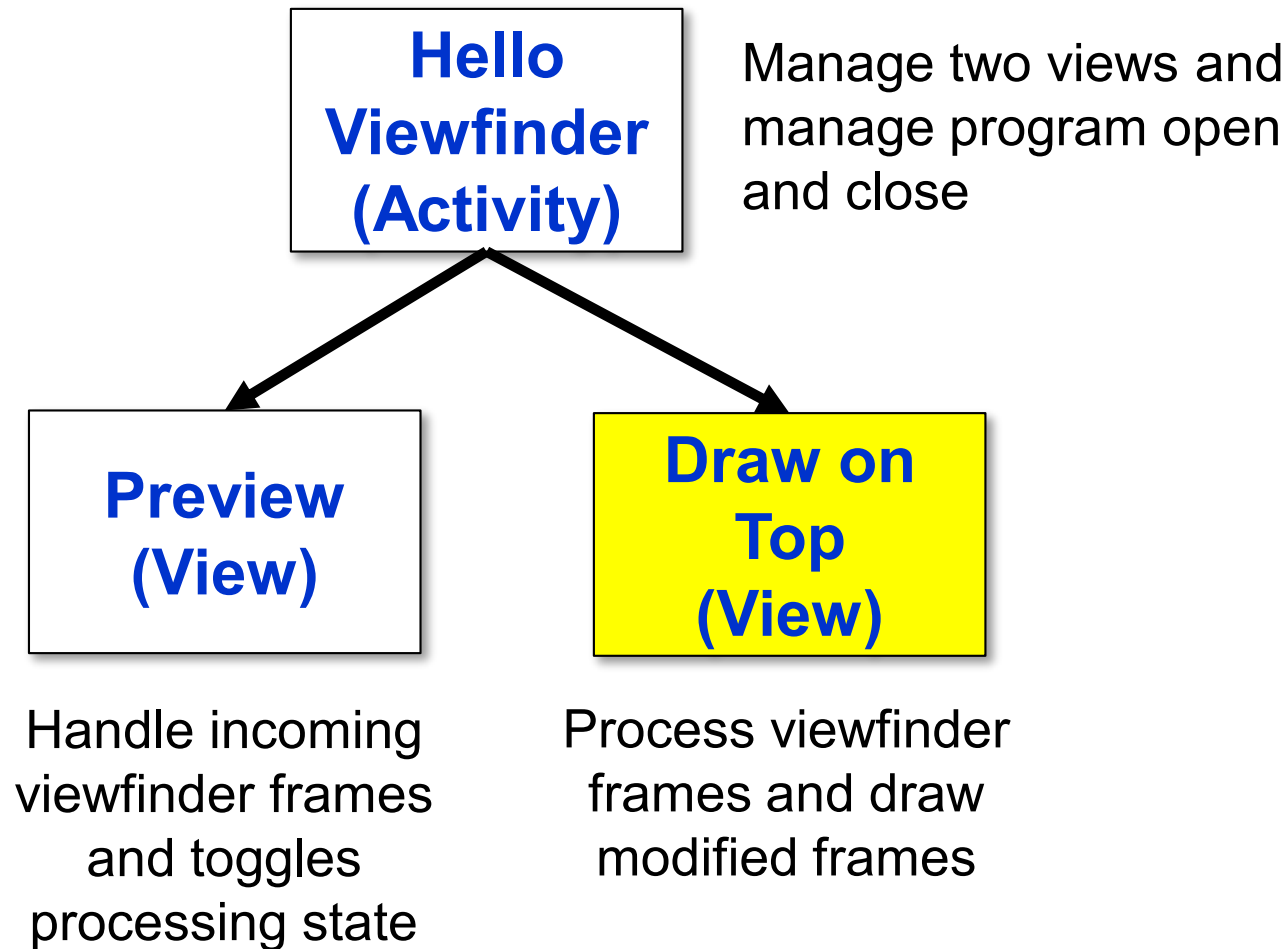
```
// Define on touch listener
this.setOnTouchListener(new OnTouchListener() {
    public boolean onTouch(View v, MotionEvent event)
    {
        if (mDrawOnTop.mState == DrawOnTop.STATE_ORIGINAL)
        {
            mDrawOnTop.mState = DrawOnTop.STATE_PROCESSED;
        }
        else if (mDrawOnTop.mState == DrawOnTop.STATE_PROCESSED)
        {
            mDrawOnTop.mState = DrawOnTop.STATE_ORIGINAL;
        }
        return false;
    }
});
```

Define an anonymous touch listener object

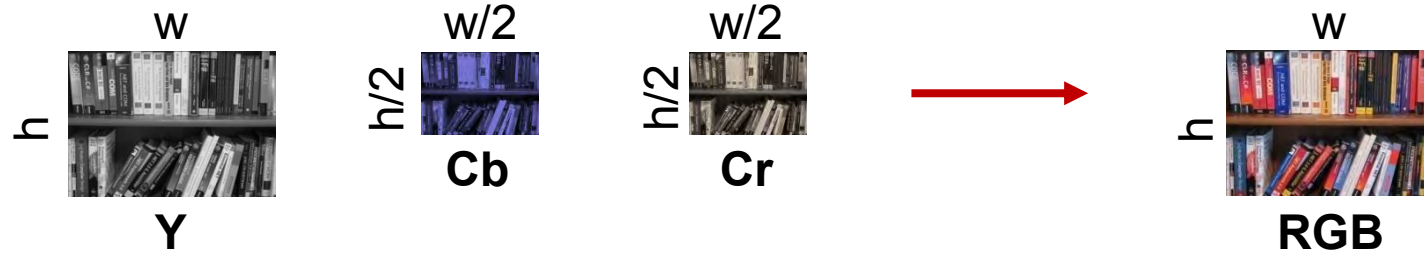
If in original state, toggle to processed state

If in processed state, toggle to original state

# “Hello Viewfinder” class hierarchy



# Draw on Top class: YCbCr to RGB conversion



$$\begin{aligned} R &= \frac{298.08 \cdot Y}{256} + \frac{408.58 \cdot Cr}{256} - 222.921 \\ G &= \frac{298.08 \cdot Y}{256} - \frac{100.29 \cdot Cb}{256} - \frac{208.12 \cdot Cr}{256} + 135.58 \\ B &= \frac{298.08 \cdot Y}{256} + \frac{516.41 \cdot Cb}{256} - 276.84 \end{aligned}$$

# Draw-on-top class: process viewfinder frames

```
// Called whenever a repaint is requested
protected void onDraw(Canvas canvas)
{
    ...
    // Convert from YCbCr to RGB
    if (mState == STATE_ORIGINAL)
        decodeYCbCr420RGB(mRGBData, mYCCData, mWidth, mHeight);
    else
        decodeYCbCr420RGBHistEq(mRGBData, mYCCData, mWidth, mHeight);

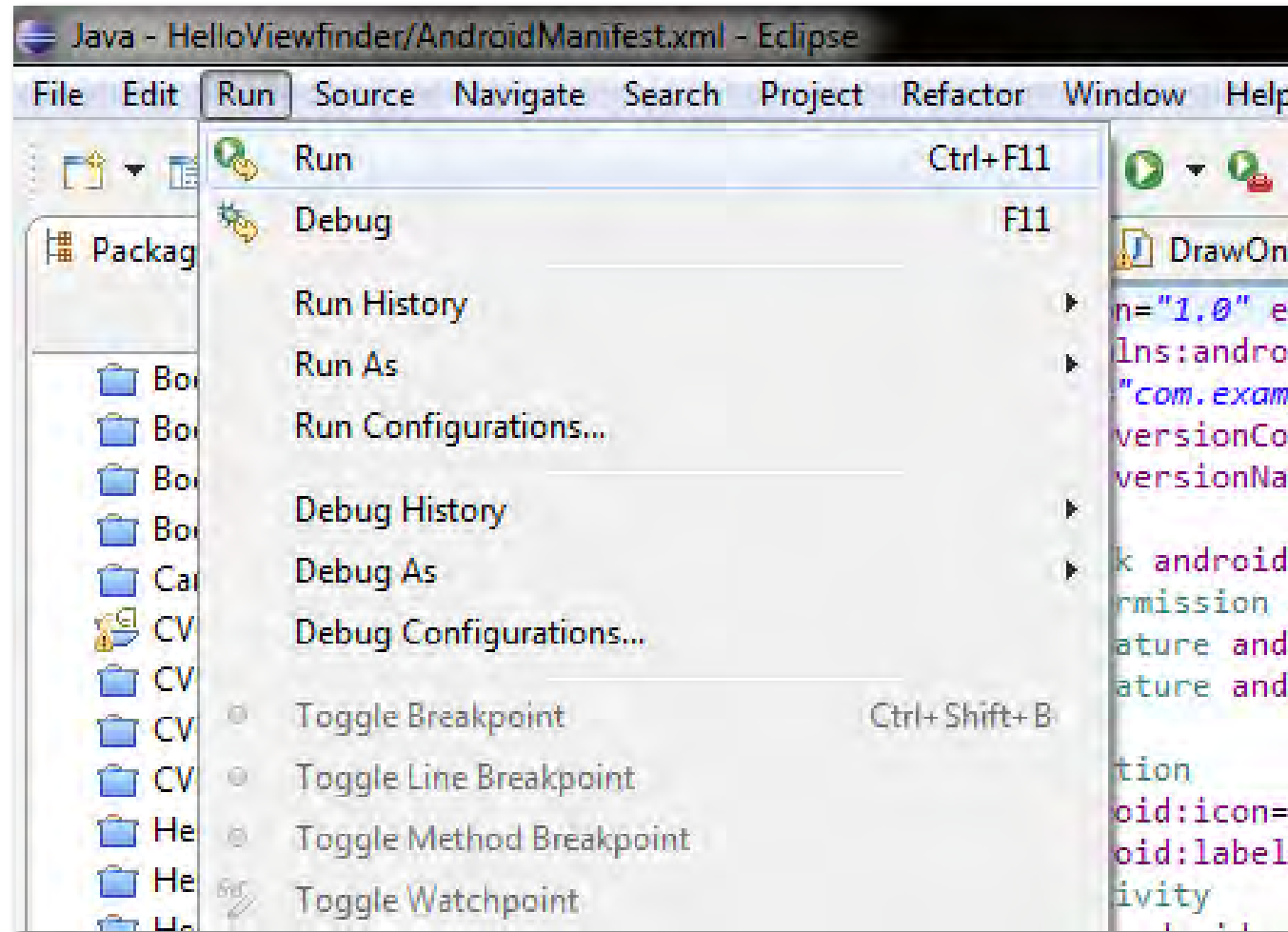
    // Draw bitmap
    mBitmap.setPixels(mRGBData, 0, mWidth, 0, 0, mWidth, mHeight);
    Rect src = new Rect(... Size parameters ...);
    Rect dst = new Rect(... Size parameters ...);
    canvas.drawBitmap(mBitmap, src, dst, mPaintBlack);
    ...
}
```

Called whenever this view is repainted

Decode frame with or without hist. eq.

Draw decoded frame in new layer

# Running the program on an Android device



# “Hello Viewfinder” application running on device

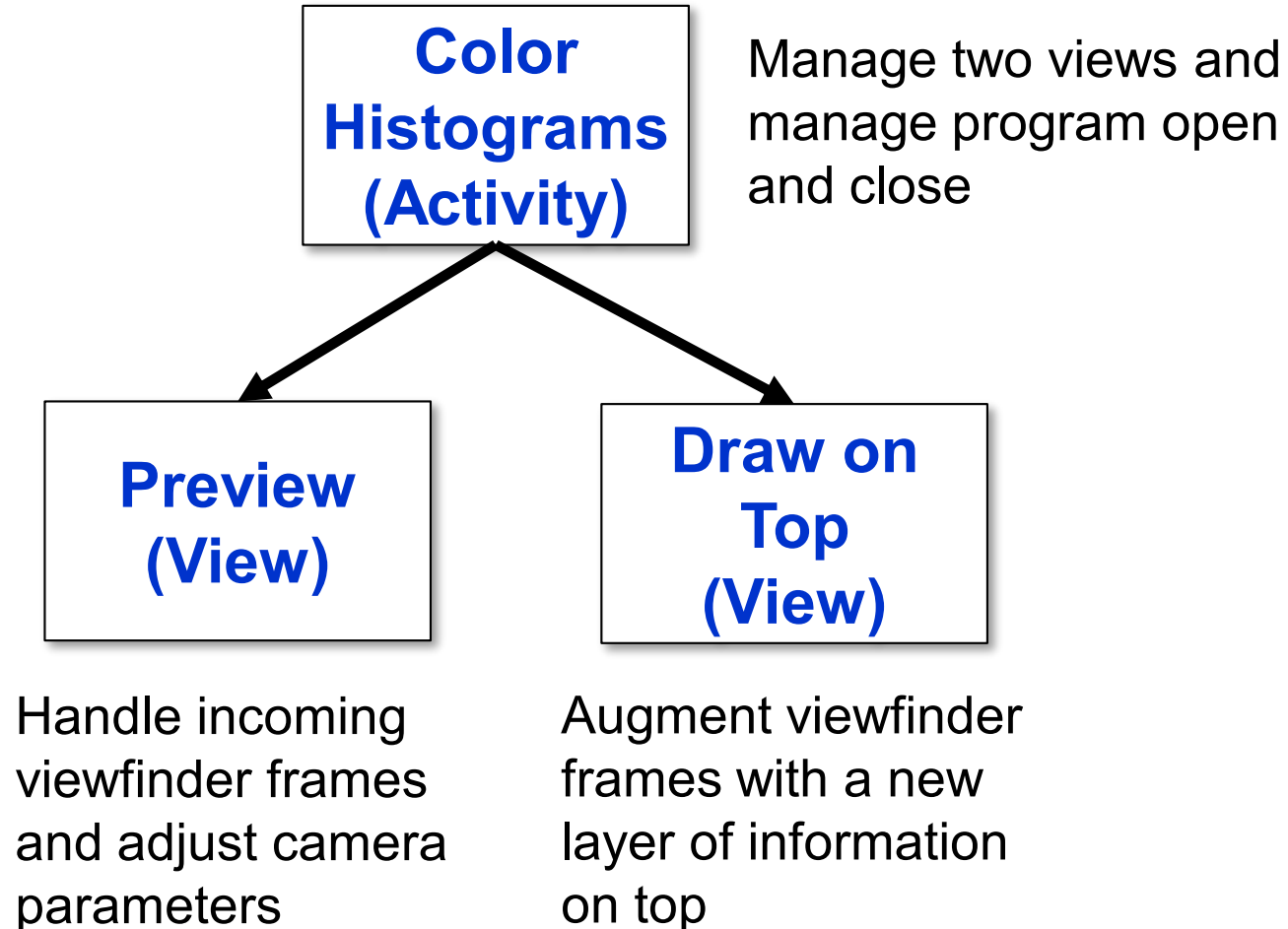


# “Color Histograms” project

- Goals of this project
  - Learn how to modify camera parameters
  - Learn how to create drawing instruments
  - Learn how to augment the viewfinder frames
- Full source available on class website

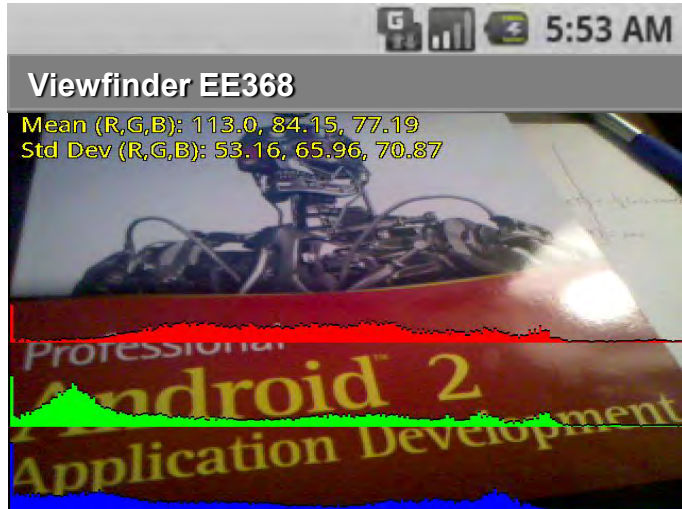


# “Color Histograms” class hierarchy



# Main activity class: set full screen mode

Icon and title bars visible

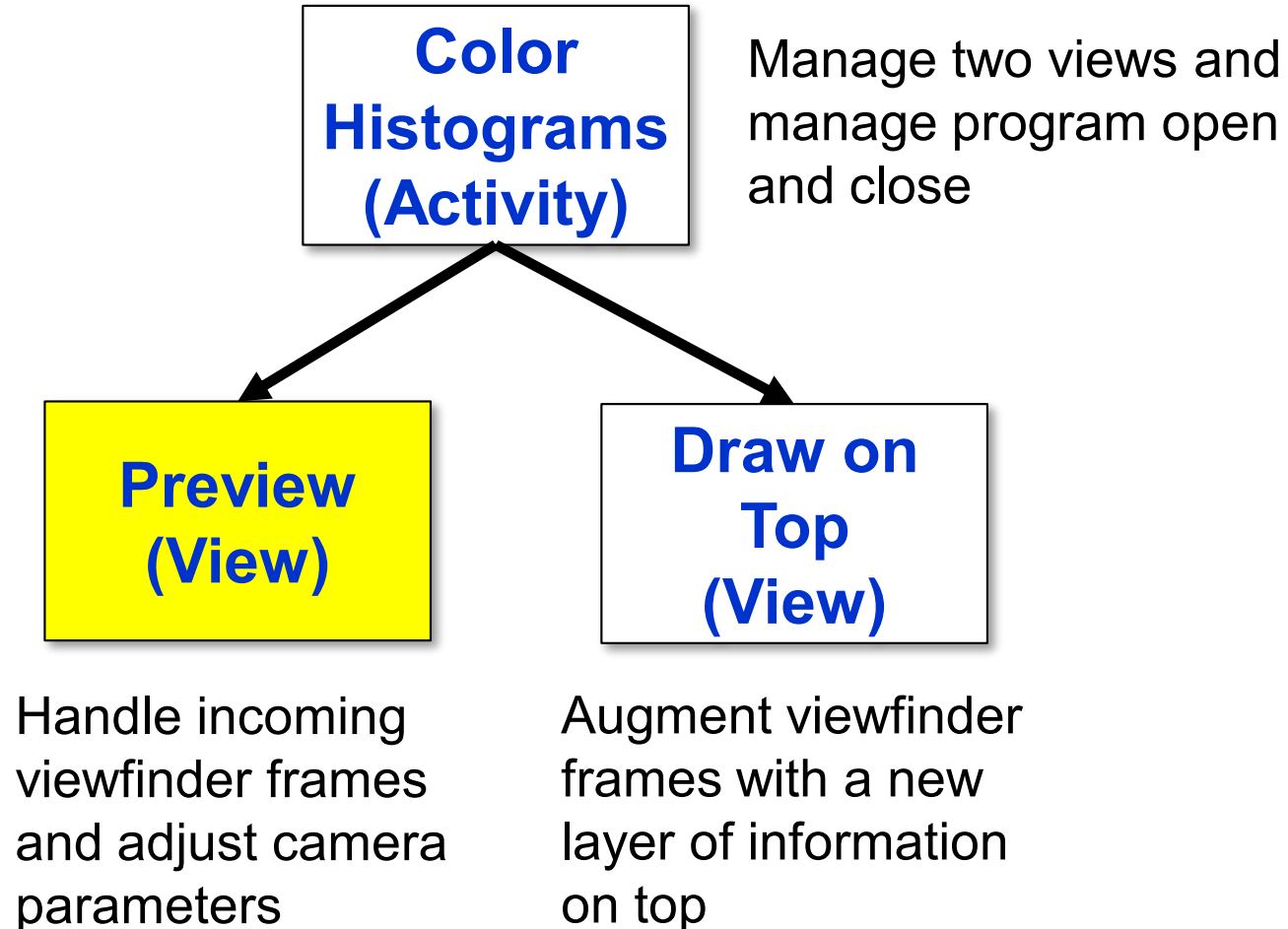


Icon and title bars hidden

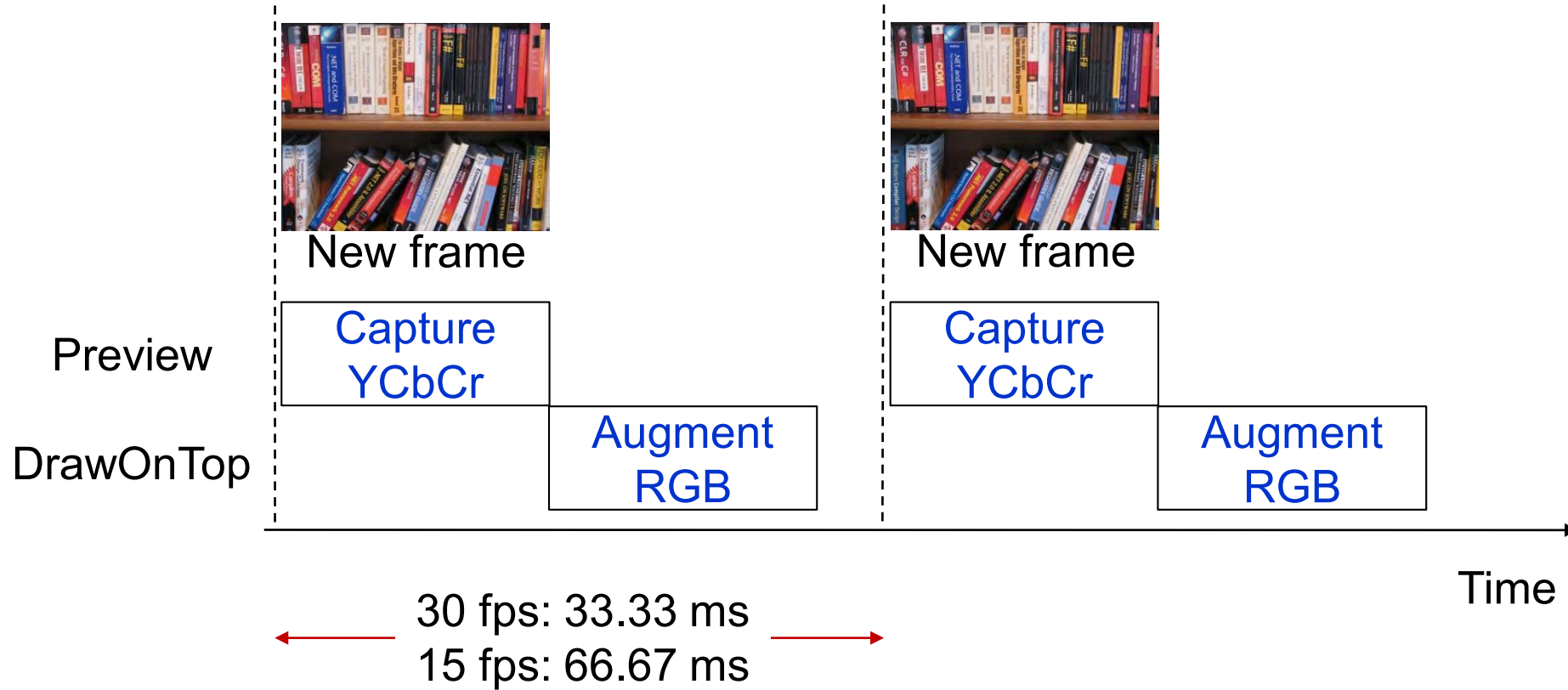


```
public void onCreate(Bundle savedInstanceState) {  
    ...  
    getWindow().setFlags(  
        WindowManager.LayoutParams.FLAG_FULLSCREEN,  
        WindowManager.LayoutParams.FLAG_FULLSCREEN  
    );  
    requestWindowFeature(Window.FEATURE_NO_TITLE);  
    ...  
}
```

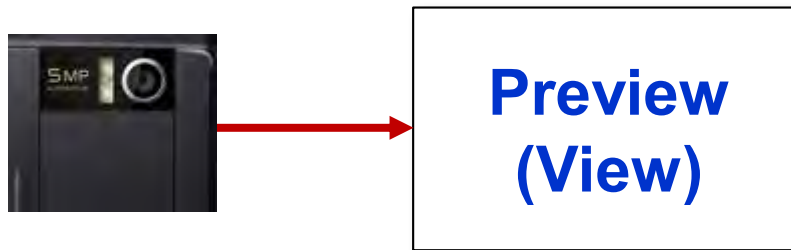
# “Color Histograms” class hierarchy



# Timeline of events on the mobile device

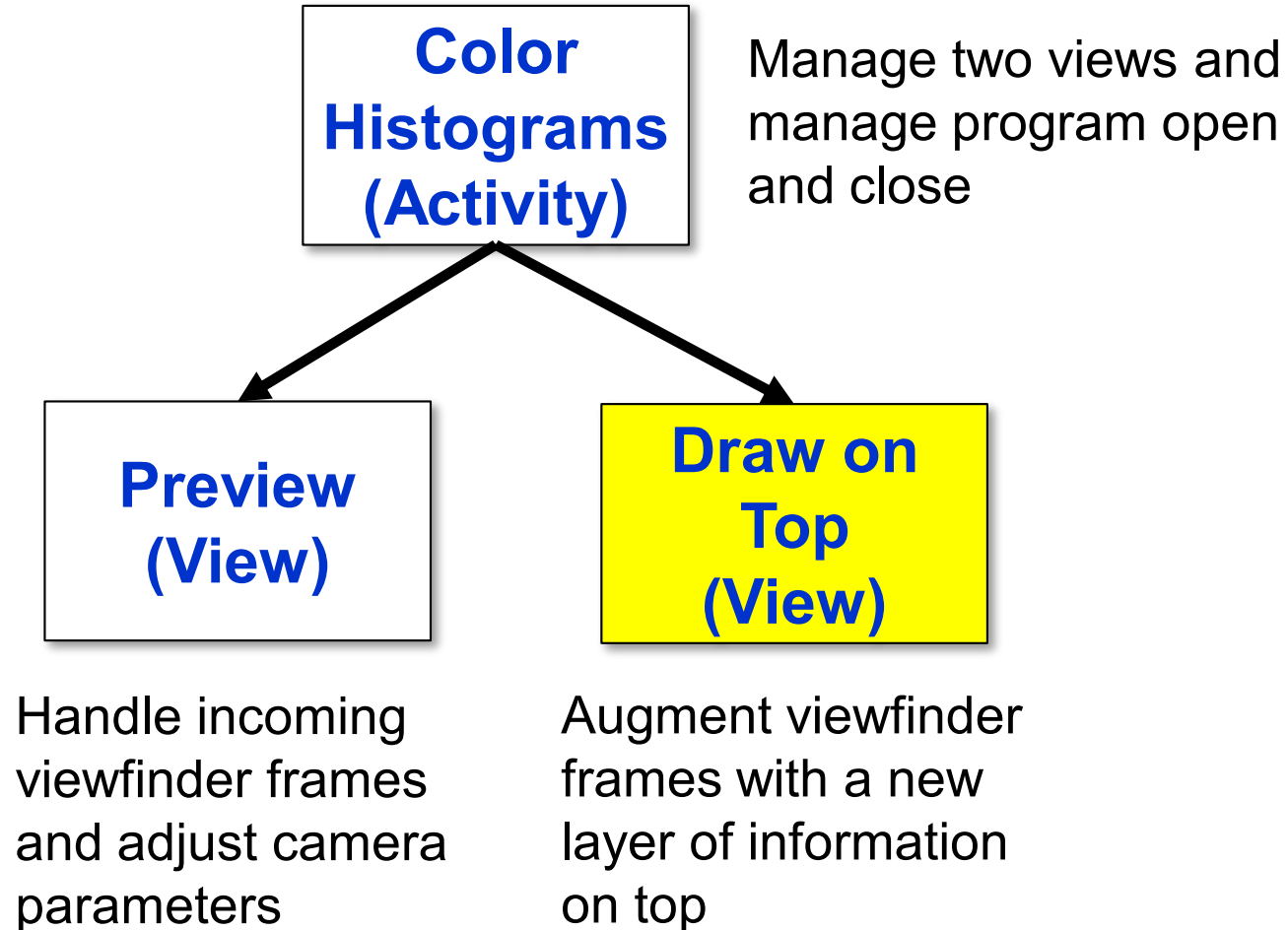


# Preview class: set camera parameters



```
Camera.Parameters parameters = myCamera.getParameters();  
parameters.setPreviewSize(640, 480);  
parameters.setPreviewFrameRate(15);  
parameters.setSceneMode(Camera.Parameters.SCENE_MODE_NIGHT);  
parameters.setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO);  
myCamera.setParameters(parameters);
```

# “Color Histograms” class hierarchy



# Draw on Top class: create some paint brushes

```
myPaintRed = new Paint();  
myPaintRed.setStyle(Paint.Style.FILL);  
myPaintRed.setColor(Color.RED);  
myPaintRed.setTextSize(25);
```

```
myPaintGreen = new Paint();  
myPaintGreen.setStyle(Paint.Style.FILL);  
myPaintGreen.setColor(Color.GREEN);  
myPaintGreen.setTextSize(25);
```

```
myPaintBlue = new Paint();  
myPaintBlue.setStyle(Paint.Style.FILL);  
myPaintBlue.setColor(Color.BLUE);  
myPaintBlue.setTextSize(25);
```



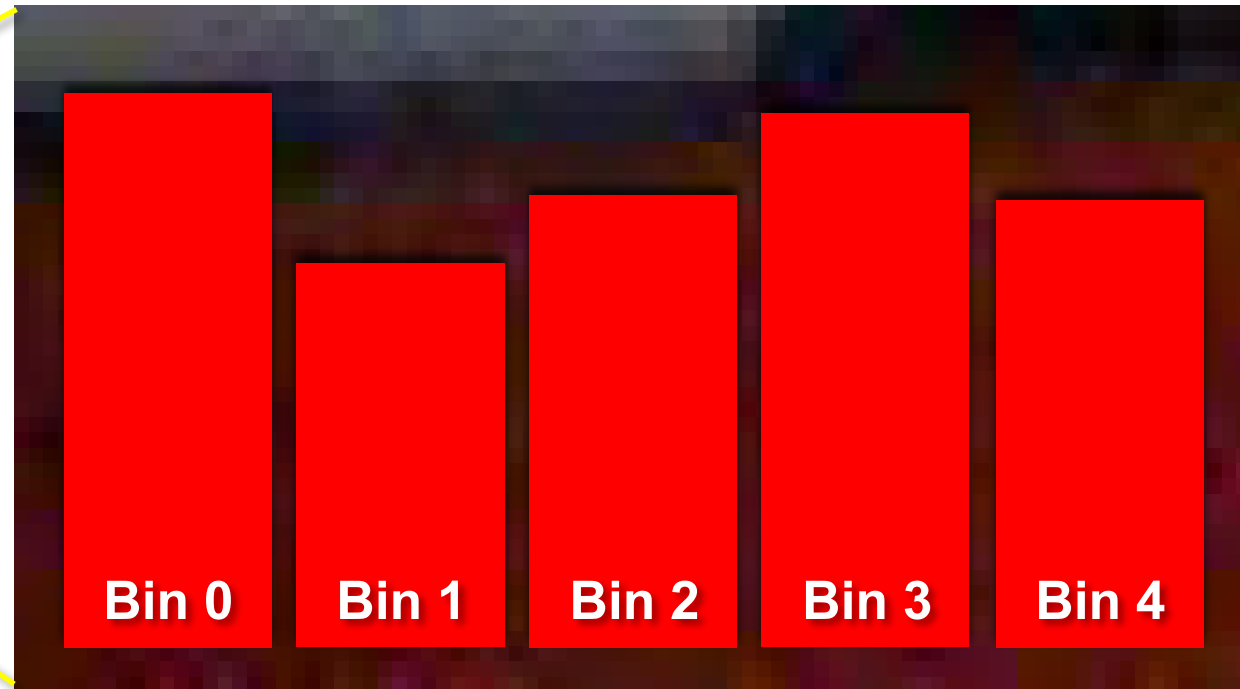
# Draw on Top class: write text on canvas



Mean (R,G,B): 113.0, 84.15, 77.19  
Std Dev (R,G,B): 53.16, 65.96, 70.87

```
String meanStr = "Mean (R,G,B): " +  
    String.format("%.4g", imageRedMean) + ", " +  
    String.format("%.4g", imageGreenMean) + ", " +  
    String.format("%.4g", imageBlueMean);  
myCanvas.drawText(meanStr, x, y, mPaintYellow);
```

# Draw on Top class: draw histograms on canvas

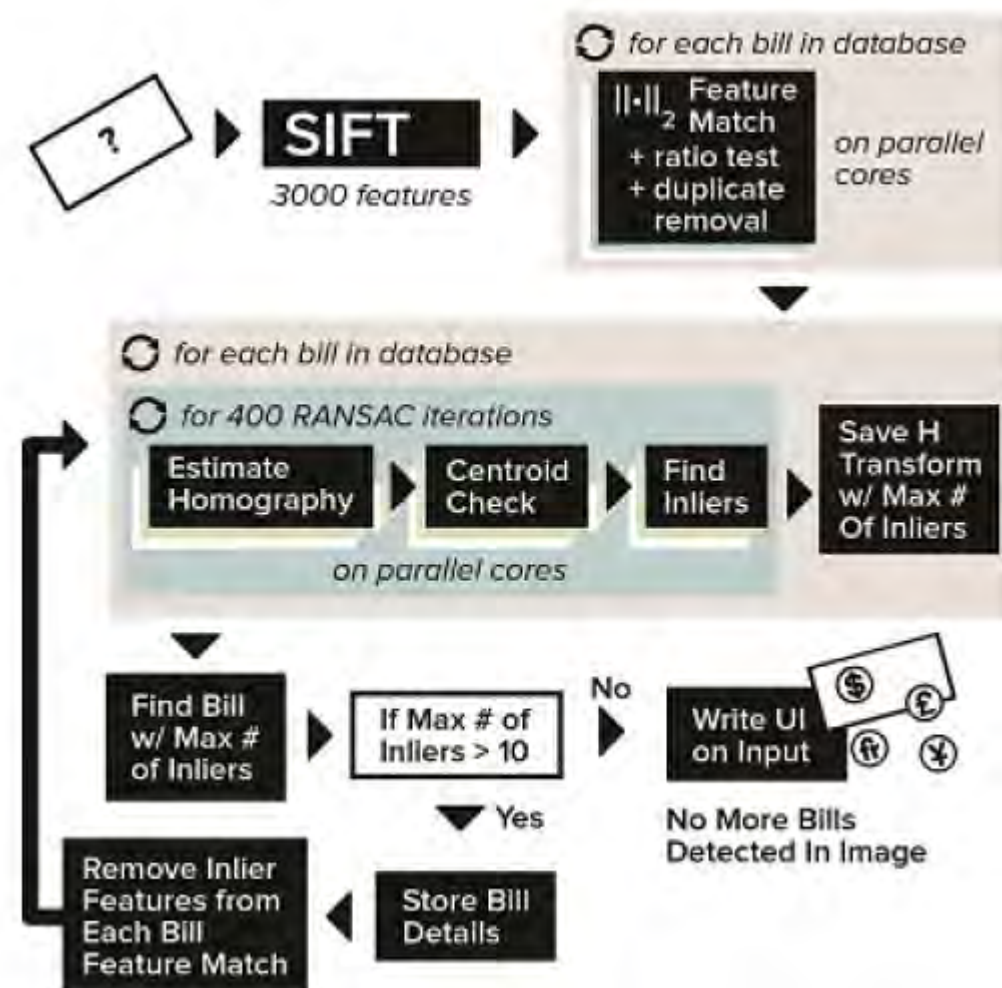


```
// Draw red rectangle  
Rect rect = new Rect(  
    left x, top y, right x, bottom y  
);  
myCanvas.drawRect( rect, myPaintRed );
```

# “Color Histograms” app running on device

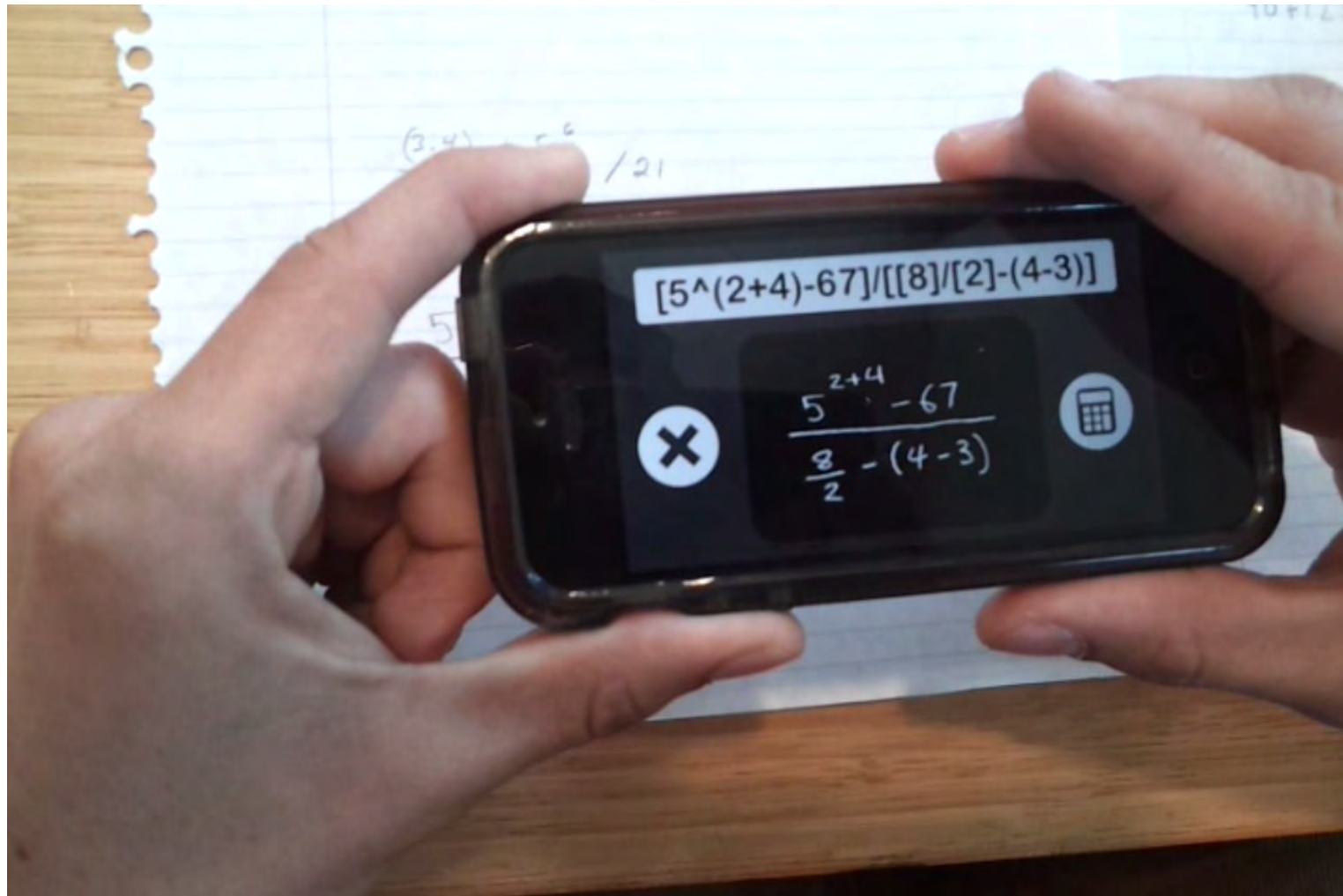


# Class project: Foreign bill detector and converter



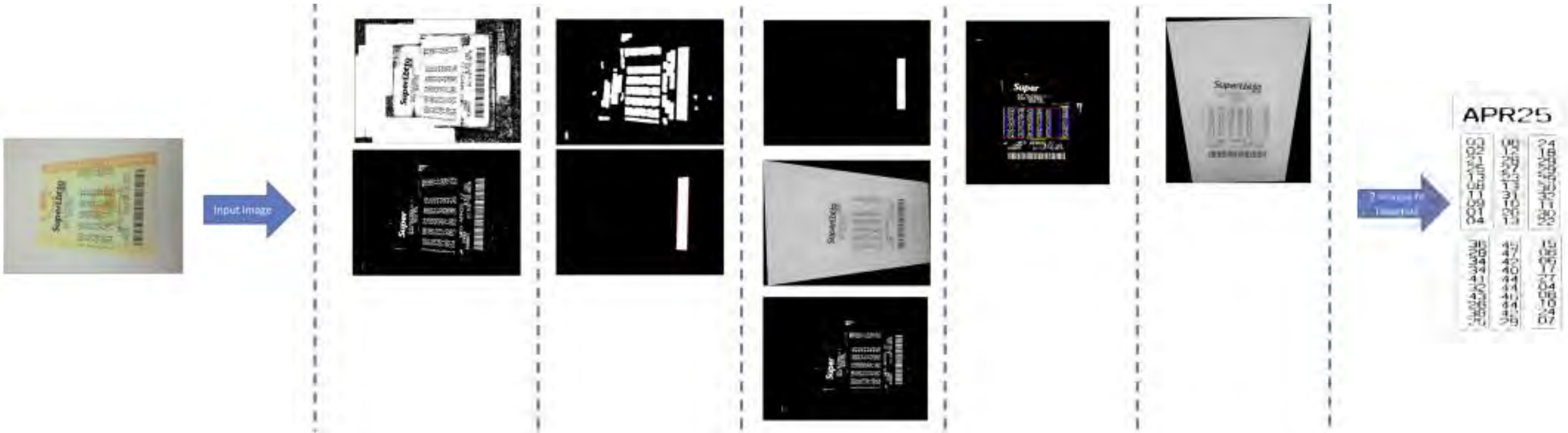
M. Digman and C. Elder, Spring 2013

# Class project: Handwritten expression solver



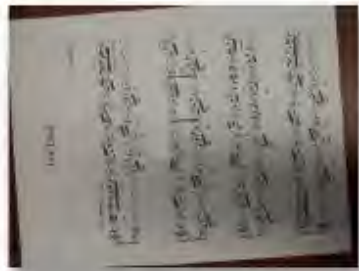
S. Harvey and W. Harvey, Spring 2013

# Class project: Lottery ticket checker



T. Zou and A. Gupta, Spring 2012

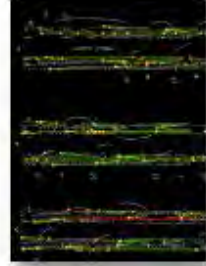
# Class project: Optical music recognition and playback



Captured Image of music staff



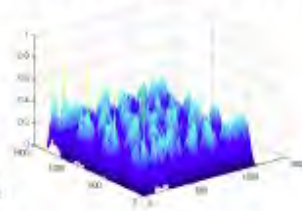
Upright Image after rotation



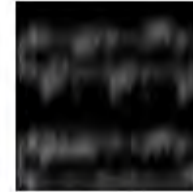
Horizontal Line Detection



Clef Template



Clef Peak Detection



Matched Clefs



Top & Bottom Staff with red marks



Detected Note Heads



S. Dai, C.-W. Lee, Y. Tian, Spring 2012