INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Rasteirinho controlled by Android

Programmer Manual

Automation Systems Course 2011/2012

Master Degree (MSc) in Information Systems and Computer Engineering

Table of Contents

Development environment requirements

- Eclipse Galielo 3.6
- JDT – Eclipse Java Development tool (http://www.eclipse.org/jdt/)

  Please note, that the latest versions of eclipse typically already include JDT, available as an update. Once Eclipse is installed, check for possible update by selecting from the menu "Help → Check for updates". Verify that JDT will be installed with the updates.
- JDK 5 or JDK 6 (JRE alone is not sufficient)
- ADT – Android Development Tools Plugin for Eclipse (http://developer.android.com/sdk/eclipse-adt.html)
- Android SDK [1]

For further details, please see http://developer.android.com/sdk/requirements.html

Install the projects:

1) cvBlobDetection

- Copy the attached project “cvBlobDetection” into the workspace.
- Import the project into the workspace (“File-->Import-->General-->Existing project into workspace” Choose the project, select “ok”)

- Verify that the project has been imported as library. In order to do this, select the project in the workspace, click with the right button and the select “Properties” as shown in figure 1

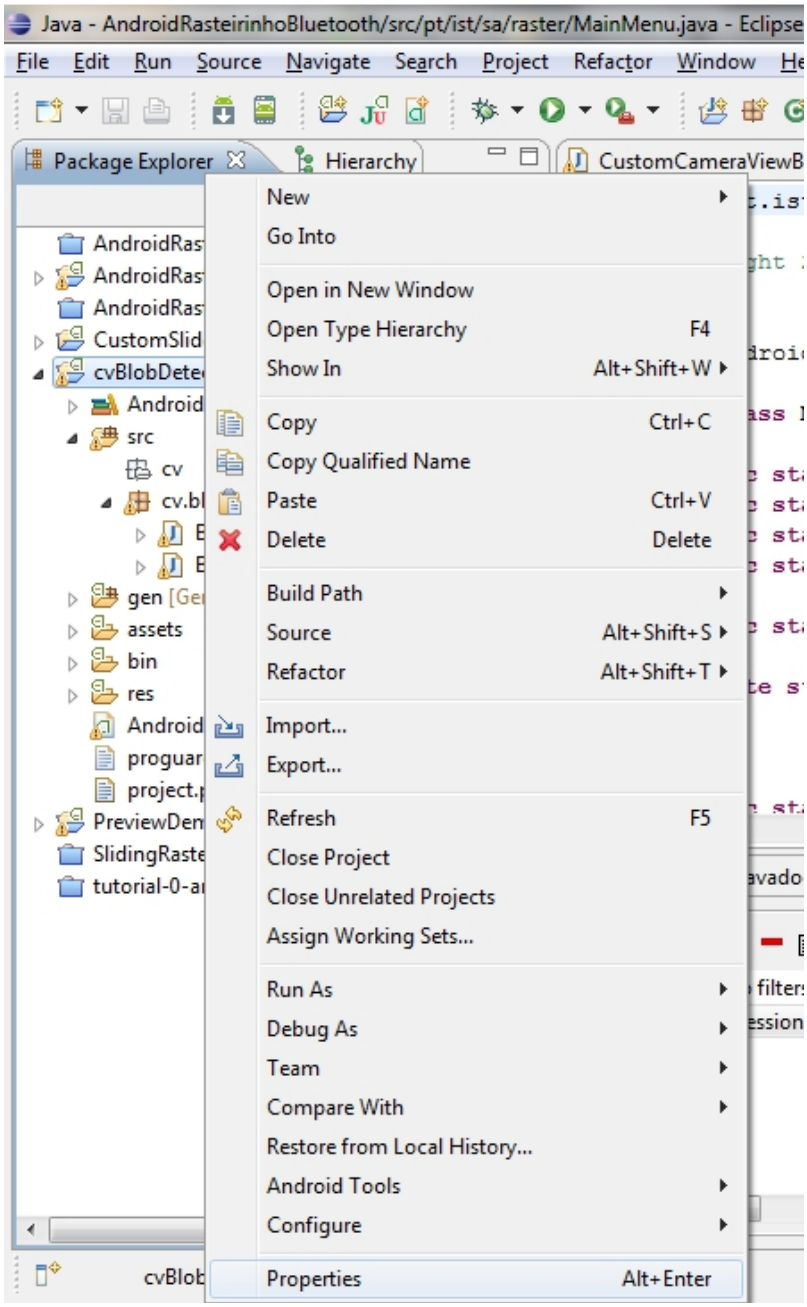

Figure 1: Project Properties

- On the left side of the pane choose “Android”. In the bottom side of the right pane verify that the checkbox “Is Library” is selected, as shown in figure 2.

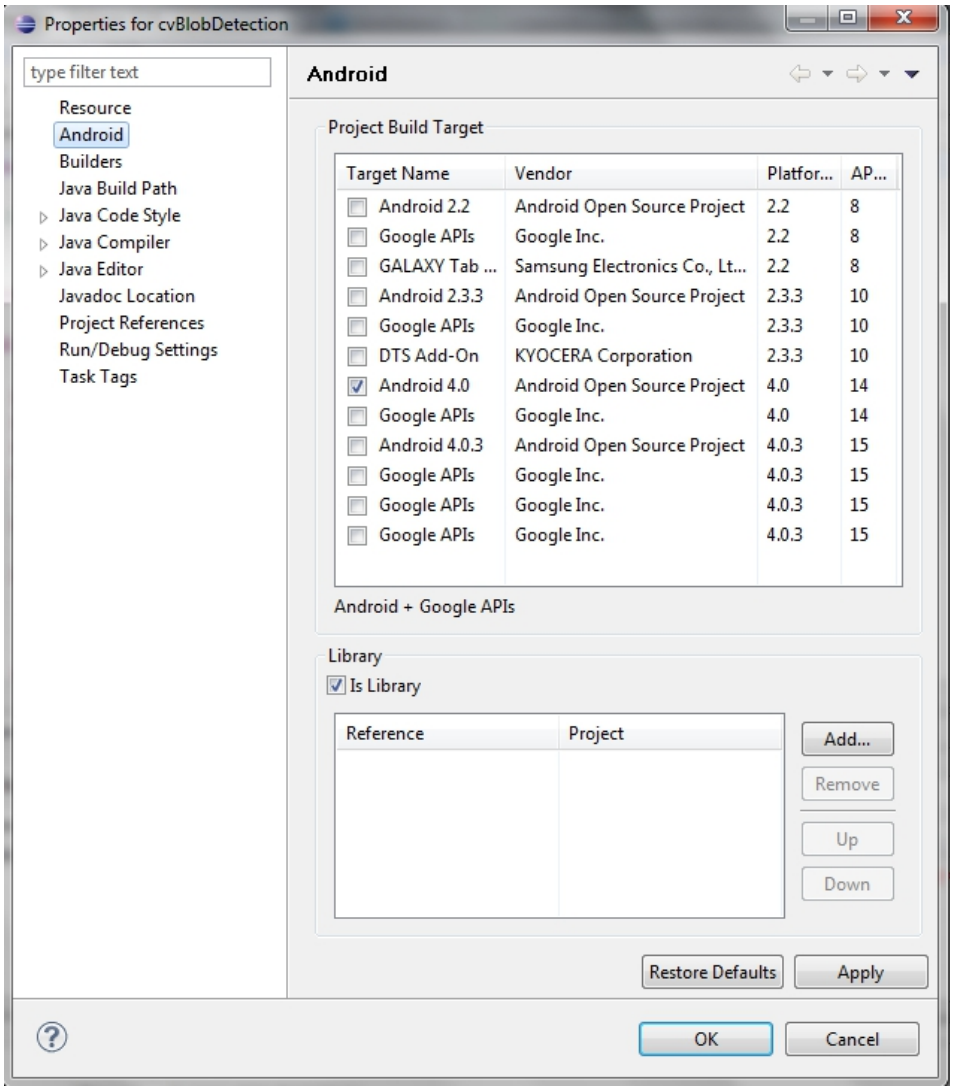

Figure 2: Specifying cvBlobDetection project as library

2) AndroidRasteirinho

- Copy the anexed project “AndroidRasteirinho” into the workspace.
- Import the project into the workspace (“File-->Import-->General-->Existing project into workspace” Choose the project, select “ok”).

- Verify that the project references correctly the project “cvBlobDetection” as library. In order to do this, select the project in the workspace, click with the right button and the select “Properties” as shown in figure 1.
-
- On the left side of the pane choose “Android”. In the bottom side of the right pane verify that the project is listed correctly, otherwise select “Add”, choose the project “cvBlobDetection”. Select “ok” to finish the procedure.

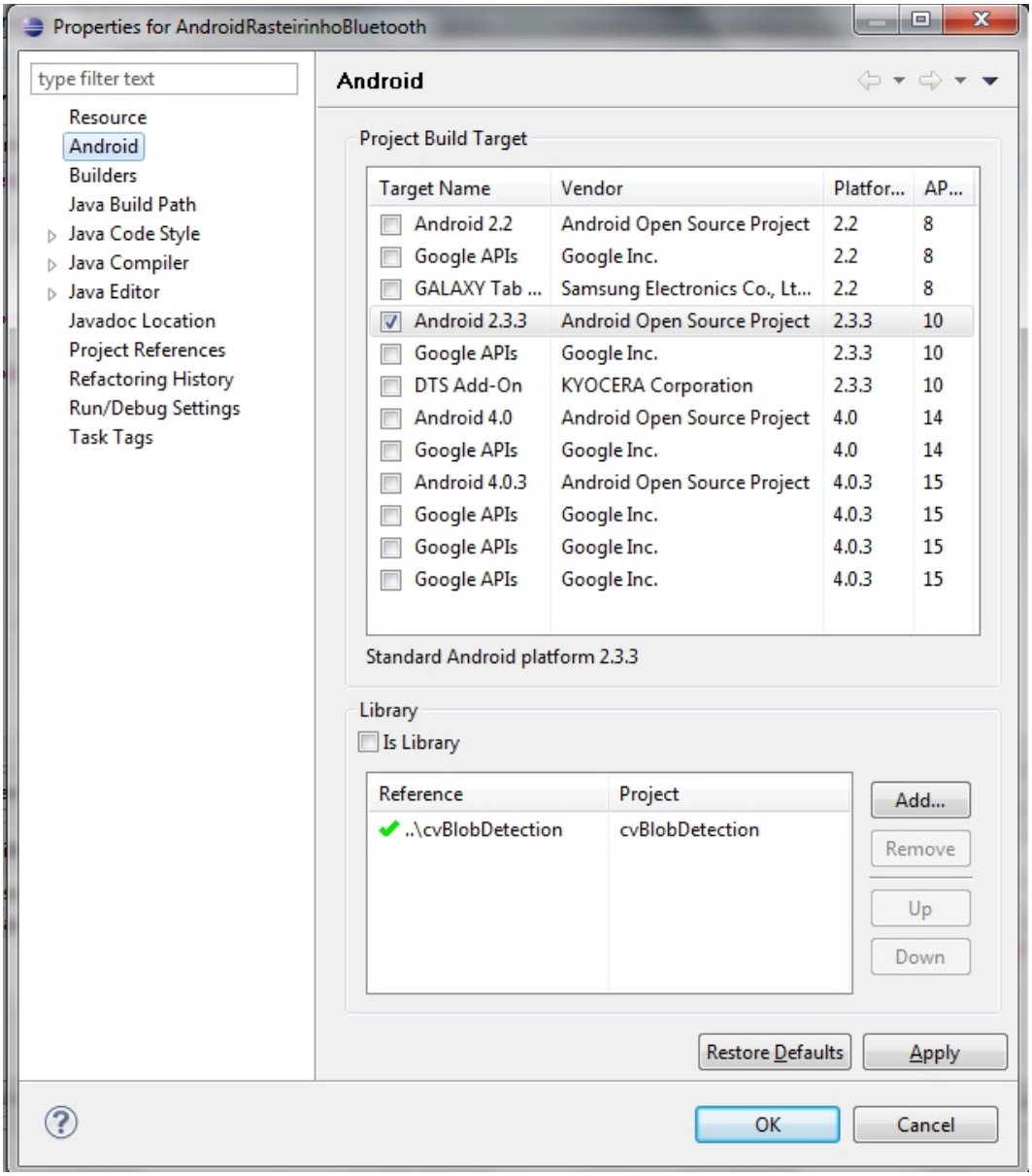

Figure 3: Specifying the use of cvBlobDetection project as library

Documentation:

1) cvBlobDetection

This library is available on google code [2] under GNU GPL 3 license.

CvBlob is a library designed specifically for Android, in order to perform blob detection and extraction. The library has been designed based on a single-pass algorithm, detecting first the blobs, and then labeling and optimizing them.

The author used as a base the Java algorithm proposed by Dr. Andrew Greensted, although he soon realized that the implementation was not suitable for Android based-phones.

The original code has been modified in order to return only the biggest blob, as reported in the following snippet:

```
if (massTable[i] >= minBlobMass && (massTable[i] <= maxBlobMass || maxBlobMass
== -1)){
      Blob blob = new Blob(xMinTable[i], xMaxTable[i], yMinTable[i],
yMaxTable[i], massTable[i]);
      if ( blobMayor == null || blob.mass >= blobMayor.mass ) {
            blobMayor = blob;
            position=i;
            XCoordMaxBlob=(blob.xMax+blob.xMin)/2;
            YCoordMaxBlob=(blob.yMax+blob.yMin)/2;
      }
}
```

Snippet 1: Code that returns the biggest blob

The original version returned a list containing all the blob found, which was not suitable for our current task.

Also, the code has been modified in order to draw the biggest blob returned.
Finally the original code deliberately ignored the blob the around the corners, in the current version these are considered.

2) AndroidRasteirinho

Class MainMenu

This class represents the starting point of the application, once it is loaded it will render a menu with the following options:

- `Preview BW`: renderizes the preview of the web cam in black and white and applies a threshold
- `Set Threshold`**:** renderizes the preview of the web cam in black and white and allows to set a value for the threshold
- `Preview Blob`: renderizes the preview of the web cam in black and white, applies a threshold and perform blob detection analysis
- `Control Rasteirinho`: besides perofrming blob detection, this option start a bluetooth comunication and controls the robot
- `Quit`: terminates the application

The layout shown in the Snippet 3 (defined under *'%projectFolder%/res/layout/main.xml'*) is responsible for rendering the afore mentioned menu.
It is loaded together with this class, as shown in part in Snippet 2:

```
/** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final Button buttonSetup = (Button)findViewById(R.id.ButtonBW);
        buttonSetup.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                        startActivity(new Intent(MainMenu.this,
                        BWActivity.class));
                  }
            });

.........
```

Snippet 2: Part of code of MainMenu class that associates to each element defined in the layout its corresponding class.

By selecting one of the afore mentioned options the corresponding *`activity`* will be loaded:
`Preview BW` will load the class `BWActivity`
`Set Threshold` will load the class `ThresholdActivity`
`Preview Blob` will load the class `BlobActivity`
`Control Rasteirinho` will load the class `ControlRasteirinhoActivity`

As shown in the class diagram (figure 4) all these *`activities`* use the class `CustomCameraView` which extends `CustomCameraViewBase.`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/linearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:orientation="vertical" >

    <Button
        android:id="@+id/ButtonBW"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Preview B/W" />

    <Button
        android:id="@+id/ButtonTrheshold"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Set Threshold" />

    <Button
        android:id="@+id/ButtonBlob"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Preview Blob" />

    <Button
        android:id="@+id/ButtonControl"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Control Rasteirinho" />


    <Button
        android:id="@+id/ButtonQuit"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Quit" />

</LinearLayout>
```

Snippet 3: Xml code that creates the menu

Class CustomCameraView

This class receives, frame by frame and in the format of an array of byte, the images captured from the webcam, applies a threshold and returns a bitmap as output.

```
@Override
protected Bitmap processFrame(byte[] data) {
      if (bmp==null){
            bmp = Bitmap.createBitmap(getPreviewFrameWidth(), getFrameHeight(),
            Bitmap.Config.ARGB_8888);
            bmp.setDensity(Bitmap.DENSITY_NONE);
      }

      for (int i = 0; i < frameSize; i++) {
            int y = (0xff & ((int) data[i]));

            if (MainMenu.viewMode!=MainMenu.VIEW_MODE_NORMAL) {
                  if (y>MainMenu.getThreshold()) y=0xff;
                  else
                        y=0;
                  rgba[i] = 0xff000000 + (y << 16) + (y << 8) + y;
            }
            else {
                  rgba[i] = data[i+getPreviewFrameWidth() *
                        BlobDetection.HOR_LINES_TO_SKIP];
            }
      }

      bmp.setPixels(rgba, 0/* offset */, getPreviewFrameWidth() /* stride */,
      0, 0, getPreviewFrameWidth(), getFrameHeight());
      return bmp;

}
```

Snippet 4: Code that processes the received bytes, applies a threshold and returns a Bitmap

Class CustomCameraViewBase

This class is the main interface with the Android Camera. As soon as the camera receives new frames, these are processed with the support of the class *CustomCameraView.*
If the activity choosen is related with the control of the robot "Rasteirinho" (ControlRasteirinhoActivity) or simply with the blob detection activity (BlobActivity) this class calls the *BlobDetection* algorithm and eventually starts the class *Commander* that control the robot via bluetooth.

```
.............
Bitmap bmp = null;
synchronized (this) {
      try {
            this.wait();
            bmp = processFrame(mFrame);
      } catch (Exception e) {
            e.printStackTrace();
      }
}

Bitmap anotherBitmap = null;

if (bmp != null) {
      Canvas canvas=null;
      canvas = mHolder.lockCanvas();
      BlobDetection.Blob b = null;
      if (MainMenu.viewMode==MainMenu.VIEW_MODE_BLOB || MainMenu.viewMode ==
            MainMenu.VIEW_MODE_CONTROL) {

            bmp=Bitmap.createScaledBitmap(bmp, 80, 60, false);
            BlobDetection blob = new BlobDetection(bmp);
            anotherBitmap = blob.getBlob(bmp);

            if (blob.blobList.size()>0){
                  b=blob.blobList.get(0);
                  currentBlobPosition = ((b.xMax+b.xMin)/2);
            }
      }

      /*DRAW THE BITMAP */
      if ( (MainMenu.viewMode == MainMenu.VIEW_MODE_BW || MainMenu.viewMode ==
            MainMenu.VIEW_MODE_NORMAL) && canvas != null) {
            canvas.drawBitmap(bmp, (canvas.getWidth() - getPreviewFrameWidth())
            / 2, (canvas.getHeight() - getPreviewFrameHeight()) / 2, null);
            mHolder.unlockCanvasAndPost(canvas);
      }

      if ( (MainMenu.viewMode == MainMenu.VIEW_MODE_CONTROL ||
      MainMenu.viewMode == MainMenu.VIEW_MODE_BLOB) && canvas != null) {
            canvas.drawBitmap(anotherBitmap, (canvas.getWidth() ) / 2,
            (canvas.getHeight()) / 2, null);
            mHolder.unlockCanvasAndPost(canvas);
      }

      /*CONTROL ROBOT */
      if (MainMenu.viewMode == MainMenu.VIEW_MODE_CONTROL && cmd.isControl() &&
            b != null) {
            int[] speedValues = getError(b,bmp.getWidth());
            if(cmd != null){
                  cmd.setV1(speedValues[0]);
                  cmd.setV2(speedValues[1]);
                  cmd.run();
            }
      }
}
.............
```

Snippet 5: Code that calls the `Blob Detection` algorithm, renderizes the Bitmap returned by `processFrame` and eventually command the robot.

Class Commander

This class starts a bluetooth communication with the robot "Rasteirinho" based on its MAC address. This value at the time of writing this document is: "`00:1D:43:00:D0:FC`"
Bluetooth communication is implemented via socket, the robot plays the role of the server, this class acts as a client, by sending command. It does not read any state from the robot.

```
public void run() {
      // Create a data stream so we can talk to server.
      if (D)
            Log.e(TAG, "+ ABOUT TO SAY SOMETHING TO SERVER +");
      try {
            outStream = btSocket.getOutputStream();
      } catch (IOException e) {
            Log.e(TAG, "ON RESUME: Output stream creation failed.", e);
      }

      byte[] mBuffer = new byte[6];
      // right wheel
      mBuffer[0] = 0;
      mBuffer[1] = 49; // set speed 1 (command 31)
      mBuffer[2] = (byte) getV1();
      // left wheel
      mBuffer[3] = 0;
      mBuffer[4] = 50; // set speed 2 (command 32)
      mBuffer[5] = (byte) getV2();

      try {
                  outStream.write(mBuffer);
      } catch (IOException e) {
            Log.e(TAG, "ON RESUME: Exception during write.", e);
            e.printStackTrace();
      }

      try {
            outStream.flush();
      } catch (IOException e) {
            e.printStackTrace();
      }
}
```

Snippet 6: Code that sends commands to the robot

Class ControlRasteirinhoActivity

This class loads a simple layout that consists only in a preview of the processed content received by the webcam. Then, sets the corresponding chosen menu option and instantiates a `CustomCameraView` to start the image processing.

```
/** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        Log.i(TAG, "onCreate");
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.simplecamera);
        MainMenu.viewMode = MainMenu.VIEW_MODE_CONTROL;

        SurfaceView preview=(SurfaceView)findViewById(R.id.preview);
```

```
        previewHolder=preview.getHolder();
        ccVb = new CustomCameraView(previewHolder);

        previewHolder.addCallback(ccVb);
    }
```

Snippet 7: Code that loads a simple layout, sets the corresponding menu option that has been chosen, instantiates a `CustomCameraView` to start image processing

Class ThresholdActivity

This class loads a different layout that besides rendering the preview of the processed content received by the webcam it also shows a sliding bar that allows to control the value for the threshold. The value is istantly updated. In order to do this, it instanties `CustomSliderView` class.

```
/** Called when the activity is first created. */
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            requestWindowFeature(Window.FEATURE_NO_TITLE);

            setContentView(R.layout.slidingcamera);
            SurfaceView preview=(SurfaceView)findViewById(R.id.preview);
            previewHolder=preview.getHolder();
            ccVb = new CustomCameraView(previewHolder);
            MainMenu.viewMode = MainMenu.VIEW_MODE_BW;

            previewHolder.addCallback(ccVb);
            CustomSliderView slider2
                  =(CustomSliderView)this.findViewById(R.id.slider2);
            slider2.setDelegateOnTouchListener(this);
            slider2.setResourceIds( R.drawable.slider_other_thumb,
                  R.drawable.slider_bar_other_skin);
            slider2.setRange(1, 300);
            slider2.setScaledValue(250);
      }
```

Snippet 8: Code that loads a sliding layout and sets the corresponding menu option that has been chosen, instantiates a `CustomsliderView` to take care of the operation related with the sliding movement.

Class BlobActivity

Similarly to class `ControlRasteirinhoActivity`, this class loads a simple layout, sets the corresponding chosen menu option and instantiates a `CustomCameraView` to start image processing.

Class CustomSliderView

This class contains utility methods that allow to change the default value for the threshold. This value is shared by all the class that perform video processing. The default value is initially set to 100.

Class Diagram

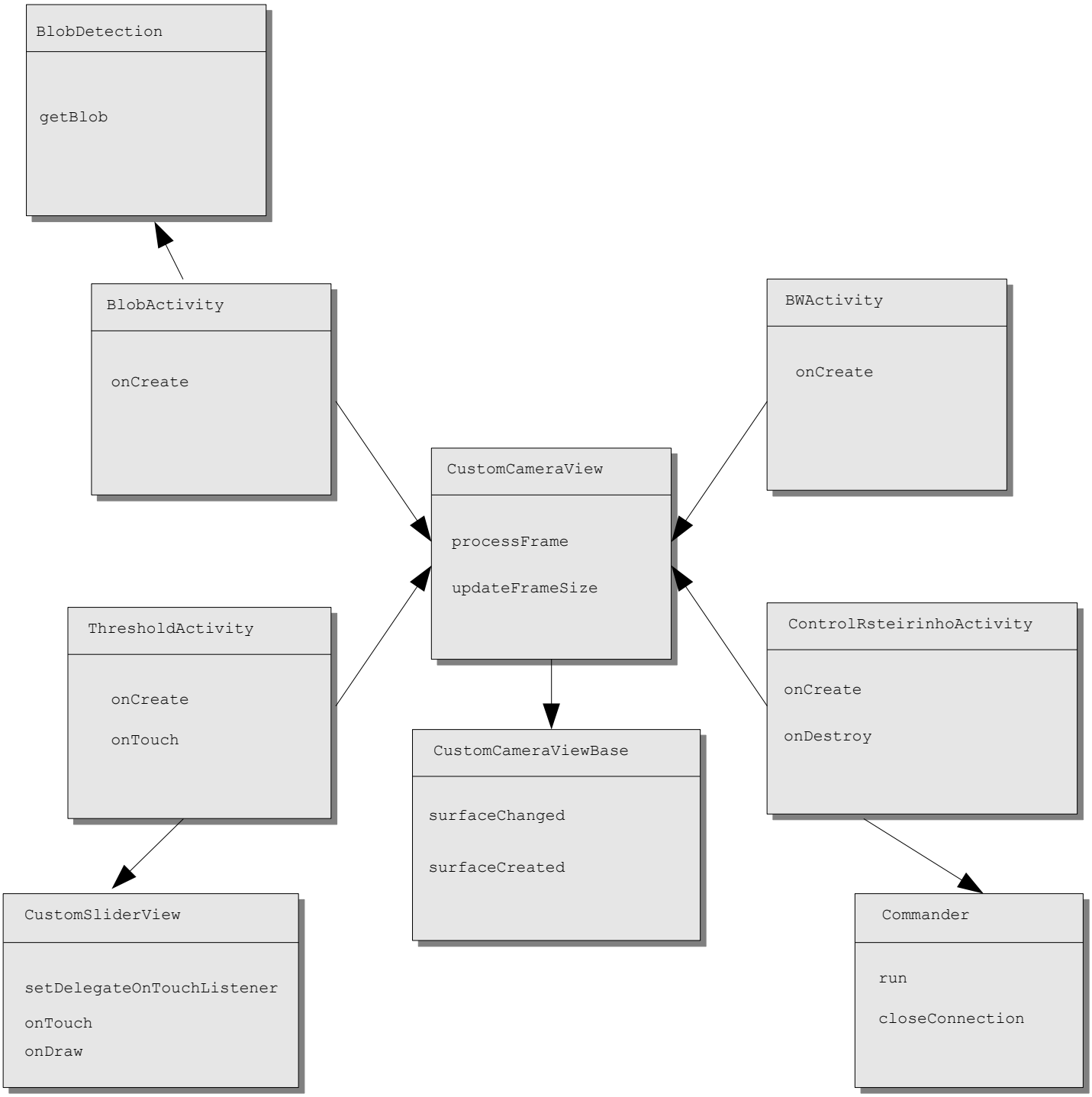

Figure 4: Class Diagram

References:

[1] Android sdk: http://developer.android.com/sdk/index.html
[2] Android cvblob: http://code.google.com/p/cvblob-for-android/