Tartu University

Faculty of Mathematics and Computer Science

Pätris Halapuu

Silver Jürimäe

Siim Puniste

Andres Traks

Conceptual Analysis of the Photo Sharing application for Android

Mobile Application Development course

Tartu 2011

[Introduction](#h.kqysythxij69)

[The home screen](#h.me9jq0vi0e2)

[Working with pictures](#h.ah9l4yf9jh8g)

[Photo capturing](#h.r2bxakj1u0sw)

[Sharing Pictures](#h.w2capgqvc1a0)

[Photo editing](#h.54a64oett5rs)

[Brightness control](#h.29g5wgrppwwn)

[Grayscale and sephia effects](#h.3bcw20tmxc3s)

[Undo](#h.pp8s88ffb11q)

[Saving the picture](#h.g81jhmivdoym)

[Memory allocation issues](#h.lpcf1s8hsss1)

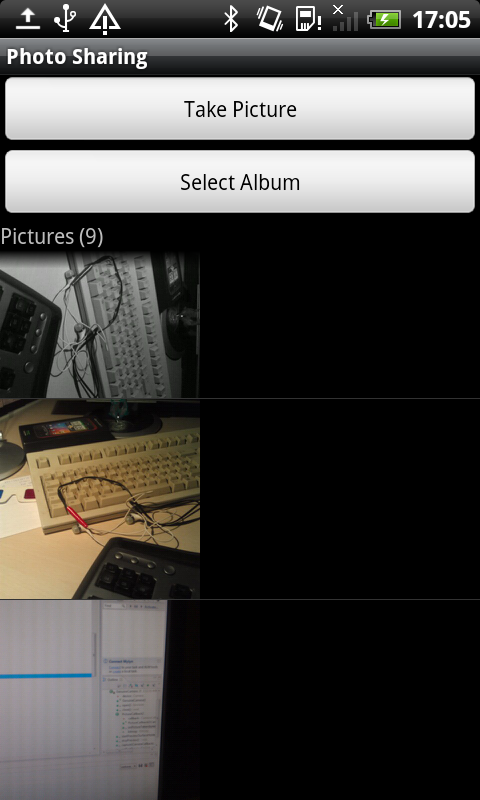
[Working with the emulator](#h.gaf0b0vqmm9q)

# Introduction

The Photo Sharing mobile application is designed to take pictures on the camera of the mobile phone, to edit the pictures and to share them with other peers using Bluetooth or other sharing methods.

The application is compatible with Android OS version 2.3.3 (Gingerbread).

# The home screen

When first running the application, the user is presented with a list of pictures on the device and two buttons, “Take Picture” and “Select Album”. In code, this screen is defined by the layout file *main.xml* and code file *PhotoSharingActivity.java*.

By default, the location of the pictures is the Pictures folder on the SD card. The user can select a different location by clicking the “Select Album” button. The device is scanned for known locations of pictures such as:

*/mnt/sdcard/Pictures/*

*/mnt/sdcard/Download/*

*/mnt/sdcard/DCIM/*

*/mnt/sdcard/downloads/bluetooth/*

In the case that the SD card is not mounted in the device, the application creates a folder in the application’s private data area and uses it as the working directory.

Thumbnails are created for the images in the list, because ImageView does not do this automatically.

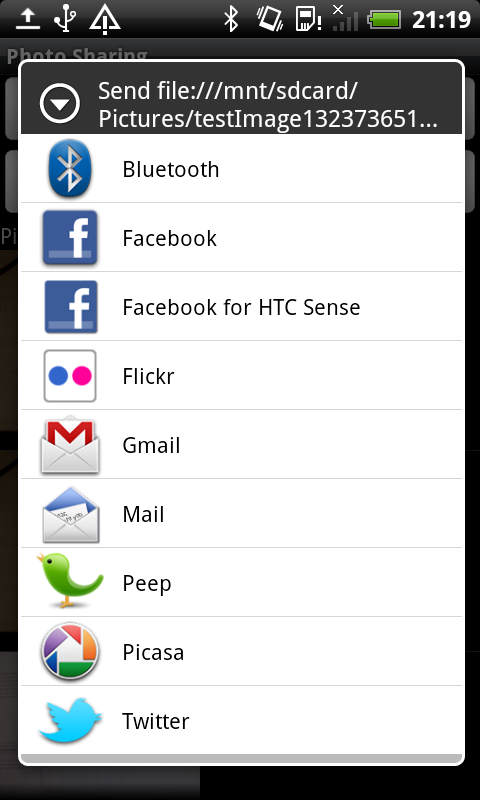
## Working with pictures

The available actions for a picture are displayed when the user touches a picture in the list (standard ListView behaviour). The options are “Edit”, “Send” and “Delete”. “Edit” starts the Editor activity with the selected picture. “Send” opens a list of options of how to share the picture, including Bluetooth and social networking sites. “Delete” deletes the picture.

## Photo capturing

The “Take Picture” button starts the *PictureCaptureActivity*. This uses the phone’s internal camera or a web camera on the emulator (with a workaround, see “Working with the emulator” below).

# Sharing Pictures

During development, several methods of sharing pictures were attempted. The task was to find a way to do peer detection and to send the picture to the peer that was found. Android provides a low-level interface for Bluetooth communication, which was used to find all the nearby Bluetooth devices.

It later became clear that Android has a more convenient way to share files over BT, the ACTION\_SEND intent. The benefits of the Send intent are that 1) it is very simple to use, all that is needed is to pass the URI and the MIME type of the image file to the intent and 2) the Send intent allows the user to also select other ways to share the image. For example, not only can the picture be shared via Bluetooth, but it can also be used as an e-mail attachment or sent to a social networking site.

The final code used to send the image:

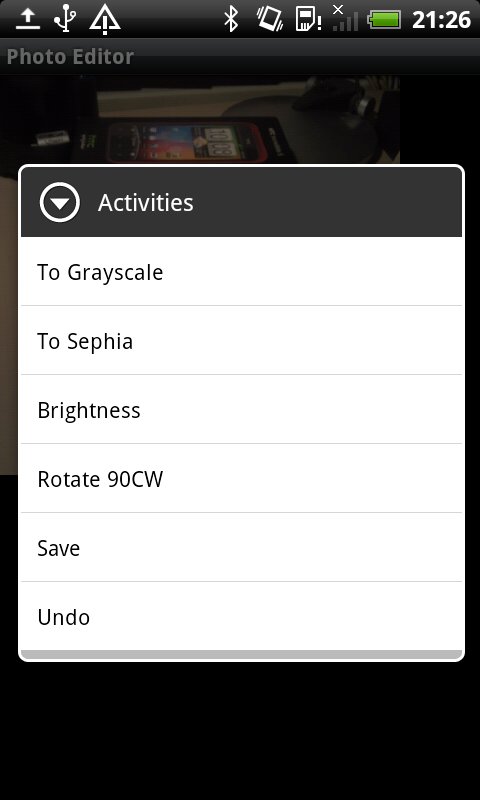
*Intent share = new Intent(Intent.ACTION\_SEND);*

*share.setType("image/jpeg");*

*share.putExtra(Intent.EXTRA\_STREAM, uri);*

*startActivity(Intent.createChooser(share, "Send " + uri.toString()));*

# Photo editing

When selecting “Edit” on a picture’s context menu, the picture is opened in another activity (PictureEditorActivity). At first, the screen only shows the picture being edited. The context menu can be activated by tapping on the picture.

The available editing actions are the grayscale and sephia effects, brightness control and rotation. When applying an effect, the image is not automatically saved, but is stored in memory.

## Brightness control

When selecting “Brightness” on the context menu, two buttons appear that can be used to increase or decrease the brightness of the picture. The activity switches to another layout with two buttons in addition to the ImageView. The current bitmap is transferred from the previous ImageView to the one in the new layout.

The brightness is adjusted using the getPixel and setPixel methods of the Bitmap. The algorithm first makes a copy of the bitmap. Then it loops over each point, fetching the ARGB value with getPixel(x,y), and increases or decreases the red, green and blue values of the pixel. Finally, the pixels are written back with setPixel(x,y) and the resulting bitmap sent to the ImageView.

## Grayscale and sephia effects

These work much the same way as changing the brightness – by looping over each pixel and changing the RGB values. However, the grayscale effect is made by modifying the bitmap through a *ColorMatrixColorFilter*[[1]](#footnote-1).

## Undo

The Editor activity maintains a list of previous image states in an array of Bitmaps. This makes it possible to implement the undo action. “Undo” can be selected from the context menu. If the device supports an accelerometer, then it is also possible to initiate an undo by shaking the phone for a short while. The algorithm compares sensor data every 100 ms to find out if the phone is being moved significantly faster than normal. On registering an undo request, the activity takes the previous bitmap from the array and sends it to the ImageView. Bitmaps that are no longer used are destroyed (redo is not implemented). The sensor management code is in *ShakeListener.java*.

## Saving the picture

The picture can be saved from the “Save” option in the context menu. A name for the newly create image file is automatically generated by appending the current time to the end of the name of the original file.

To simplify dealing with saving and loading of image files, a static class called *MediaManager* is used. In addition to having simple methods to save and load images, this class also looks for appropriate locations to store images and maintains the current working directory.

## Memory allocation issues

In cases where the editable bitmap is very large (more than 3-4 Mpx), the application may fail with an out of memory error. Since the undo list maintains all the previous picture states, then an error may also occur with a large number of edits.

# Working with the emulator

Since Android phones were not available for all team members, extra steps had to be taken to ensure that the application could be run and tested also on the emulator. This involves augmenting two components that the emulator lacks – the camera and Bluetooth connectivity.

The camera part was solved by using an additional component that downloads a web camera feed over the internet.[[2]](#footnote-2) When the application detects that it is running on the emulator, then this option is used.

We found no workaround to get Bluetooth to work in the emulator, but since sharing pictures was delegated to Android’s own ACTION\_SEND intent, then this wasn’t a problem.

1. <http://developer.android.com/reference/android/graphics/ColorMatrixColorFilter.html> [↑](#footnote-ref-1)
2. <http://www.tomgibara.com/android/camera-source> [↑](#footnote-ref-2)