ITIS/ITCS 4180/5180 Mobile Application Development
Homework 3
Date Posted: 06/11/2014 at 14:00
Due Date: 06/15/2014 at 23:55

Basic Instructions:

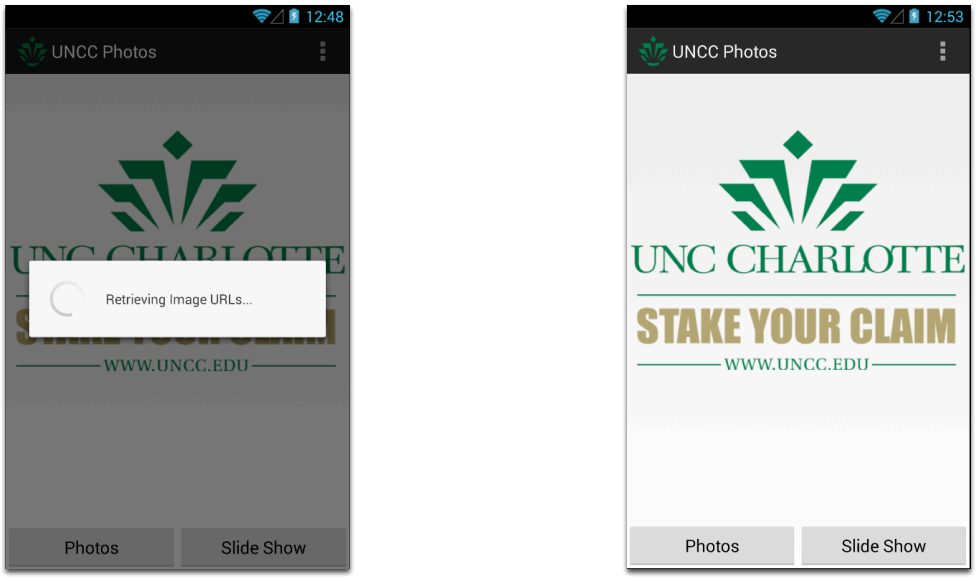
1. In every file submitted you MUST place the following comments:
   a. Assignment #.
   b. File Name.
   c. Full name of all students in your group.
2. Each group should submit only one assignment. Only the group leader is supposed to submit the assignment on behalf of all the other group members.
3. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
4. Please download the support files provided with this assignment and use them when implementing your project.
5. Export your Android project as follows:
   a. From eclipse, choose *"Export…"* from the File menu.
   b. From the Export window, choose *General* then *File System*. Click *Next*.
   c. Make sure that your Android project for this assignment is selected. Make sure that all of its subfolders are also selected.
   d. Choose the location you want to save the exported project directory to. For example, your *Desktop* or *Documents* folder.
   e. When exporting make sure you select *Create directory structure for files*.
   f. Click Finish, and then go to the directory you exported the project to. Make sure the exported directory contains all necessary files, such as the .java and resource files.
6. Submission details:
   a. When you submit the assignment, compress your exported Android project into a single zip file. The format of compressed file name is HW#.zip
   b. You should submit the assignment through Moodle: Submit the zip file.
7. **Failure to follow the above instructions will result in point deductions.**

Homework 3 (100 Points)

In this assignment you will develop a very simple UNC Charlotte photo gallery application for android. The app is composed of 2 activities, namely **Main Activity**, and **PhotosActivity**. The provided support files includes a single stake your claim photo. In this assignment you will get familiar with multi-activity communication using Intents, and multi-threading.

Important App Requirements:

1. The required Android Virtual Device (AVD) should have **minimum SDK version set to 14 and target SDK at least 17**. The app should display correctly on 3.2" QVGA (ADP2) (320x480: mdpi). **Your assignment will not be graded if it does not meet these requirements, and you will not be granted any points on your submission.**
2. All strings should be read from your strings.xml, all dimensions from dimens.xml, and all images from drawable-ldpi. The string values used for the text labels, and button labels should be read from the `strings.xml` file and should not be hardwired in the layout file.
3. All image downloading should be performed using Threads (or AsyncTask) and your code should not block the main thread.



(a) Main Activity retrieving the image urls. (b) Main Activity after retrieving the image urls.

Figure 1. Main Activity UI

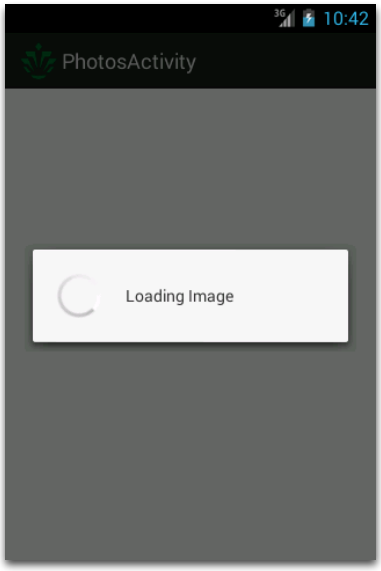
Part 1: Main Activity (20 Points)

The **Main Activity** is a simple activity that contains two buttons. The interface should be created to match the user interface (UI) presented in Figure 1(a). The implementation requirements include:

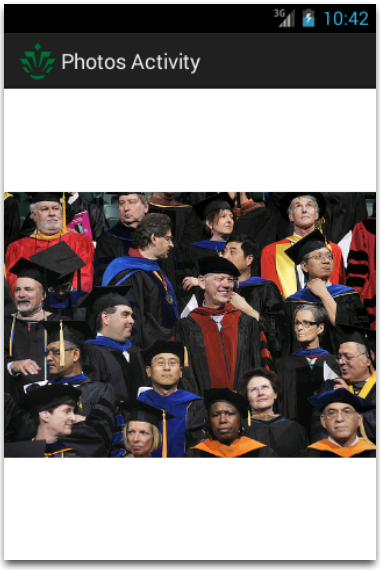
1. Your application should have an application launcher icon, please select your

launcher icon to represent your app. Set the application title to “Photo Gallery”.

2. When the activity starts it should contact the following URL to retrieve the list of photo urls, http://liisp.uncc.edu/~mshehab/api/photos.txt. You should use a Thread or AsyncTask to download the list of photo urls. A non-cancelable ProgressDialog should be displayed, and should be dismissed after the photo urls are downloaded as indicated in Figures 1(a) and 1(b). This activity should only download the photo urls indicated in the url above, the activity should not download the actual photos.
3. Tapping the “Photos” button should start the Photo Activity in photo display mode. The list of photo urls should be sent to the Photo Activity, using the intent. You will not be given any points if you use a static variable to share data between activities.
4. Tapping the “Slide Show” button should start the Photo Activity in the slide show mode. The list of photo urls should be sent to the Photo Activity, using the intent. You will not be given any points if you use a static variable to share data between activities.



(a) Photo Activity Loading Image



(b) Photo Activity Showing Loaded Image

Figure 2, Photo Activity

Part 2: Photo Activity in Photo Display Mode (30 Points)

The **Photo Activity** is started in the Photo Display Mode by the Main Activity when the user taps the “Photos” button. The list of photo urls should be sent from the Main Activity to the Photo Activity using the intent extras. In this mode the Photo Activity should start by displaying the first full size image based on the image url provided in the list of photo urls. See Figures 2(a) and 2(b). The implementation requirements include:

1. Use a thread pool (or AsyncTask) to download the photo. While the photo is being downloaded display a ProgressDialog indicating that the photo is being loaded, see Figure 2(a). Do not use the main thread to download any photos.
2. The ProgressDialog should not be cancelable. The ProgressDialog should be

dismissed after the photo is downloaded and displayed.

3. The images should be downloaded only when needed, your implementation should not pre-download all the images. At any given time only maintain a reference to the displayed photo. Do not store the downloaded images in memory as this is not efficient and will crash your application.
4. Use the finger swipe direction on the screen to decide which full size image to be displayed next. If the user swipes from left to right, this should display the next full sized photo. For example, if the currently displayed image has an index value of 5 in photo url list, then swiping from left to right should display the image with the index value 6. If the index value of the currently displayed image is N-1, then swiping from left to right should rollover and display image 0, where N is the size of the list holding the photo urls.
5. Similarly, If the user swipes from right to left should display the previous full sized photo. For example, if the currently displayed image has an index value of 5, then swiping from right to left should display the image with the index value 4. If the index value of the currently displayed image is 0, then swiping from right to left should rollover and display image N-1, where N is the size of the ArrayList holding the photo urls.

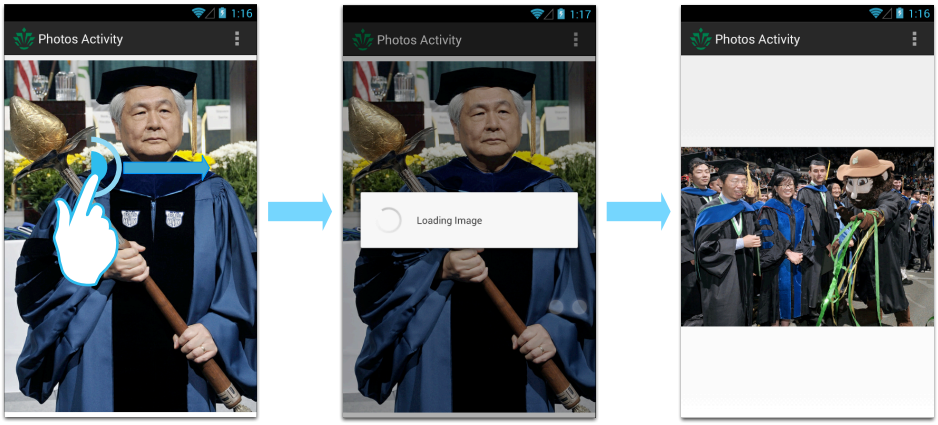


Figure 2(a) Swipe right to display next image

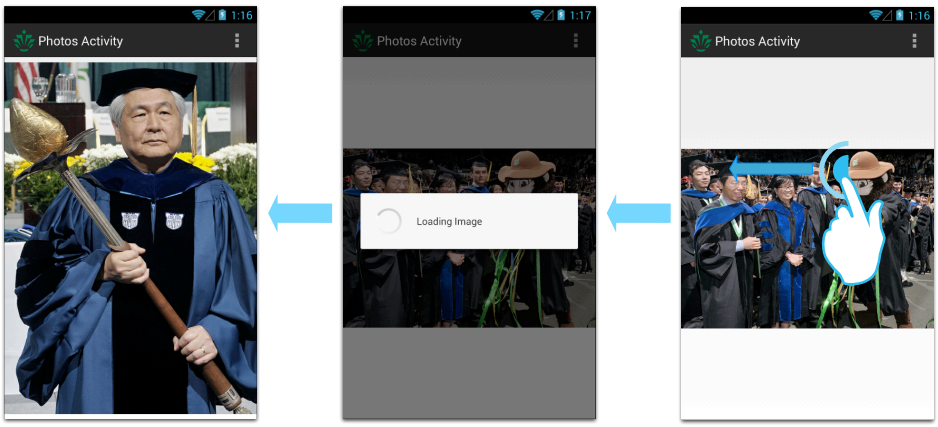


Figure 2(b) Swipe left to display previous image

Part 3: Photo Activity in Slide Show Mode (30 Points)

The **Photo Activity** is started in the Photo Display Mode by the Main Activity when the user taps the "Slide Show" button. The list of photo urls should be sent from the Main Activity to the Photo Activity using the intent extras. In this mode the Photo Activity should start by displaying the first full size image based on the image url provided in the list of photo urls. In this mode the photos should be displayed automatically in sequence with a 2 seconds delay between a photo being displayed and the next photo being displayed. The implementation requirements include:

1. Use a thread pool (or AsyncTask) to download the full sized photo. Do not use the main thread to download any photos. While the photo is being downloaded **do not** display a ProgressDialog.
2. In this mode, the swipe left/right indicated in Part 2 should be disabled.
3. The images should be downloaded only when needed, your implementation should not pre-download all the images. At any given time only maintain a reference to the displayed photo. Do not store the downloaded images in memory as this is not efficient and will crash your application.
4. If the index value of the currently displayed image is N-1, then after waiting 2 seconds rollover and display image 0, where N is the size of the array holding the image urls.

Part 4: Caching (20 Points)

Improve the efficiency of the image loading by implementing a disk cache to store the downloaded images. Before downloading an image you should check the cache and only download it if the image is not present in the cache otherwise load it from the cache. Your implementation should use the disk cache described in the android developers: http://developer.android.com/training/displaying-bitmaps/cache-bitmap.html. Integrate your image cache in the Photo Activity to enhance it's efficiency.