ECE 1778 - Creativity and Programming for Mobile Devices
October 2016
Programming Assignment P4

Threads, Local Databases and Face Detection

Mobile applications will often need to categorize and store data in a local database. This may be for purposes of caching the most recently used data for quick access, or for storing app data never intended to be stored on a server, or for prototypes apps in this class where a web backend is not required to advance the core idea. and a local database may be used as a proxy.

The use of background *threads* is also vital in many apps to perform long-running tasks. These are kept in the background and so off of the main UI thread in order to keep the user interface fluid and responsive. Background tasks can include things such as complex signal processing, optimization, or network communication.

In this assignment you will be creating a Google photos-style application which will import photos from the standard photos directory into a local database and determine which of these photos contain faces using a pool of background threads. A large portion of the UI and list management code should be reusable from Assignment P3 if that assignment was developed in a modular way.

1 Local Databases

Both Android and iOS support databases using SQL in which users are required to create the database schema and manage conversions from database queries to runtime objects (and vice-verse). In this assignment, however, you will be using a cross-platform object oriented database library called Realm. Realm allows you save and retrieve objects directly, query those objects based on the status of its member variables (for example: query all People objects whose member variable age is greater than 65) and will transparently save any modifications of objects returned through queries.

Documentation for Realm for all platforms including how to setup your Android studio or Xcode project to use Realm can be found at https://realm.io/docs/

2 Reading

Read the following if you are developing on Android:

i. Chapter titled "Dealing with Threads" of the **The Busy Coder's Guide to Android Development**, version 7.6..
ii. Chapter titled "Progress Indicators" of the **The Busy Coder's Guide to Android Development**, version 7.6.
iii. Realm Documentation, https://realm.io/docs/

iv. Google play services face detection:
http://android-developers.blogspot.ca/2015/08/face-detection-in-google-play-services.html

Read the following sections from the course texts, if you are developing on iOS.

Object C - Beginning iPhone 7 Development Exploring the iOS SDK
Swift: Beginning iPhone Development with Swift 2

i. Chapter 15 ("Grand Central Dispatch, Background Processing, and You").
ii. Realm Documentation, https://realm.io/docs/
iii. Apple face detection using the coreimage library 'CIDetector', https://developer.apple.com/library/content/documentation/GraphicsImaging/Conceptual/CoreImaging/ci_detect_faces/ci_detect_faces.html

3 Assignment

NOTE: As in previous assignments, when writing your code for this assignment, please be sure to follow 'Braiden Brousseau's Guide To Quality Apps' that was given as part of Assignment P1. Part of your grade will be assigned based on following those guidelines. However, this will be limited to the 'import' screen for this assignment as the gallery and single image views will have already been graded in the previous assignment'

You are to write an application that will allow the user to import and view photos from the devices' DCIM folder, quickly sort photos based on the presence of faces and draw boxes around each face that appears in a photo when viewed individually. The app will have only 3 primary screens: an import screen, an image gallery screen and a single image-viewing screen

Screen 1: Importing Photos by Populating the Local Database

- This screen will be used to populate the local realm database with 'photo' objects. Each photo object should contain a path to the image file and a list of face regions (x, y, width, height) found in that image.
- The app should launch to this screen first if the local realm database has no data currently stored in it, otherwise the app should launch to the image gallery screen, screen 2, described below.
- This screen should have a button to start a database import process. Pushing this button should cause any images in the DCIM folder which are not currently in the realm database to be added to the database. This will involve loading each required image and running the Android/iOS face detector on it.
- This import process *must* be set to happen on Background threads, it must not be done of the UI main thread. There should be an option to select how many concurrent threads should be simultaneously importing photos. These options should include 1, 2, 4, and 8 concurrent threads.

- During the import process some form of progress indicator or dialog should let the use know how many of the photos have been imported and how many are still left to import.
- There should be a button on this page to allow the complete deletion of all entries in the database (for testing / marking purposes). Note that this should not delete the actual jpg pictures from the DCIM directory, only the database entries.
- When the import process is done the app should switch back to the image gallery view.

Screen 2: Gallery Screen

- The primary purpose of this screen is to display image thumbnails in a gallery-style grid view (as was the case in assignment P3)
- The gallery should have 2 modes, one to display all photos referenced in the Realm database (which were already imported) and a second that will only display photos which contain faces. The difference between these modes can be accomplished with a different database queries.
- When a thumbnail in the gallery is pressed it should launch a new screen where the image can be seen in its full size and resolution.
- It should be possible to navigate from this screen to back to the import screen.

Screen 3: Single Image Screen

- This screen's primary purpose is to view a large version of the photo selected by the user
- It should be possible from this screen to delete the photo (that is remove it from the local Realm database)
- If the photo being display contains one or more faces a rectangular box should be drawn around each face.

Video Demo
You can find a video demo of the functioning application here:
https://drive.google.com/file/d/0B4IUVtTor8XHaHVqRXpkazdVSzA/view?usp=sharing

Hints

When developing this app it is recommended that you begin by fully completing the assignment using just 1 concurrent background thread for the import process. After that works you should attempt to extend it to 2, 4 and 8 threads. When extending to multiple concurrent threads, you will find that naive implementations quickly run out available memory and potentially leading to a crash. A little careful thought will be required to avoid have unnecessary copies of data, images or processing objects in each of the threads. Also be aware that even if an image in the DCIM folder from the camera is 10 or 12 megapixels you don't need that much detail to run the face detector. Images roughly the size 1280x720 (720p) are more than large enough for the detection of faces, and use a lot less memory!

Due date: Thursday November 3rd, 6pm, marked out of 10, 0.5 marks off every hour late. Submit your solution on Blackboard portal, **Programmer Assignments** link and **Assignment P4**.

What to submit:

1. Android developers: a zip file containing your final Android application file (.apk); use your student number as the filename. Also submit the complete Android Studio project directory in a separate zip file.
2. iPhone developers: you must submit the complete project directory, including source, in a zip file. Use your student number as the filename. Please do your development on the 8.0 version of the SDK, and make sure that you haven't included any files by reference. In fact, please test your submitted zip file before sending it in.