Got Movie?

A bar code reading and movie cataloging application.

By Justin Chrysler

Table of Contents

1. Introduction 3
2. Goals 3
4. Reading UPC-A Bar code 4
5. Application Internals 6
  5.1 Bar code reader 6
    5.1.1 Difficulty with Android camera 7
    5.1.2 Picture Quality 7
    5.1.3 Lighting 8
    5.1.4 Memory 9
    5.1.5 Types of Thresholding 9
  5.2 User Interface 11
  5.3 Database 12
  5.4 Web Services 12
    5.4.1 Web Services – Looked into but not used 13
    5.4.1 Web Services – Used 13
6. Conclusion 14
7. References 15

1. Introduction

The Android operating system (OS) has become one of the most cost effective platforms for independent developers to create smart phone applications. The open source nature of the OS allows developers full reign to develop any kind of application they wish. With this in mind, I decided an application that can read bar codes of movies and catalog the movies in an internal database would be an excellent way to learn this powerful OS and application programming interface (API).

There are currently other programs for the Android OS that do this already, although they each have their own limitations. The most common limitation is that the free editions limit you to a certain number of movies that you can store. These other applications also do not have an easy way to mark movies that you have lent or borrowed nor the ability to add a movie to a wish list category.

2. Goals

The initial goals of the project were to have the ability to read all of the most popular one dimensional product bar codes. This includes UPC[1]-A, UPC-E, EAN[2]-13 and EAN-8 bar codes. In theory this isn't a difficult task as UPC bar codes are actually a subset of EAN bar codes. They only vary in the number of digits encoded. Due to time constraints and lack of test products for EAN bar codes they were not implemented. The majority of products in the United States have UPC-A bar codes so this was the most important for me to get working in the application at this time.

Another feature in the initial goal was to develop a way of sharing your movie catalog with friends. This social aspect was to be used as a way of not only synchronizing with a remote server for back up purposes but to enable a way for friends to see what movies you have, what

[1] Universal Product Code

[2] International Article Number – Renamed from European Article Number

movies you want and what movies you are currently borrowing from other friends. This would be an excellent feature for future versions of this application but due to time constraints it was not implemented.

Developing the ability to read UPC-A bar codes took longer than expected so as mentioned above a few of the initial goals had to be cut. The application does however have the functionality to categorize movies that are scanned. These categories allow the user to store movies in whatever fashion they like. There are four default categories; Unsorted, Wish List, Loaned and Borrowing that cannot be removed from the system. In the movie listing functionality the user also has the ability to sort their collection on any of the information stored about the movies.

4. Reading UPC-A Bar code

UPC-A bar codes are separated into four areas; the number system being used, the manufacturer code, the product code and the check digit. The number system is the first digit and indicates what type of product the bar code represents. Movies, groceries and other commercial products are generally marked by either a 0 or a 7 for this digit. Other values include; 3 for drug/health items, 4 for in-store non-food items and 5 for coupons. The check digit is used to verify the bar code is read correctly.

The manufacturer and product codes both consist of five digits each. The manufacturer code is assigned to a company by the Uniform Code Council (UCC). The five digits allow for 99,999 possible manufacturers to be assigned codes. To allow more manufacturers with in the encoding scheme the UCC has begun using variable-length manufacturer codes. This allows for a manufacturer to be assigned more than five digits. This affects the number of products that a particular manufacturer can use before having to acquire another manufacturer code from the UCC.

Just as with manufacturer codes the product code section allows for 99,999 possible products. This five digit value can be assigned freely by the manufacturer. As mentioned above, if a manufacturer is assigned a variable-length manufacturer code then they will have less possible product values. For instance if they are assigned a nine digit manufacturer code then they can have a maximum of nine products because their manufacturer code requires four of the digits usually used by the product code.

When looking at an UPC-A bar code you will notice three sets of bars that extend either partially or fully below the other bars. These sets of bars are known as the guard pattern. The guard pattern helps in determining where the barcode is located as well as split the bar code in half. The guard patterns on both ends of the bar code are represented by a single bar, a single space and another single bar. The center guard pattern is just the normal guard pattern surrounded by a single space on each side.

Digits in a bar code are a mix of bars and spaces. A single bar or space is known as a module. Each digit has seven modules that represent a pattern that is used to decode the bar code. How they are read is determined by which side of the middle guard pattern the digit is on. The digits on the left side have an odd parity (space, bar, space, bar) while the ones on the right have an even parity (bar, space, bar, space). Odd parity just means that the modules are in a pattern of spaces, bars, spaces followed by bars. For instance a zero is encoded in odd parity as three spaces, two bars, a single space followed by a single bar (see Table 1).

Left Digits Odd Parity S B S B	**Right Digit Even Parity B S B S**
3 2 1 1	3 2 1 1
2 2 2 1	2 2 2 1
2 1 2 2	2 1 2 2
1 4 1 1	1 4 1 1
1 1 3 2	1 1 3 2

1 2 3 1	1 2 3 1
1 1 1 4	1 1 1 4
1 3 1 2	1 3 1 2
1 2 1 3	1 2 1 3
3 1 1 2	3 1 1 2

Table 1

The last digit of an UPC-A barcode is the check digit. This value is calculated by the other digits in the bar code and is used for verification that the code was read correctly. The formula is fairly simple and involves four steps. First add all the odd digits and multiply the sum by three. Second add all of the even digits excluding the check digit. Third add the odd sum and even sum together and calculate modulo ten of the result. If the result is zero skip the next step. Fourth subtract the previous result from ten. If this result is the same as the check digit then the bar code has been read correctly.

5. Application Internals

Android applications are written in Java using the Android Software Development Kit (SDK). This SDK allows full access to the hardware of the device running the OS. This allows you to do things like take pictures, detect if there is an internet connection, access to the keyboard and much more. This project only used one external third party library which is an XML RPC Client used for communication with web services. The project also implements a very basic user interface that could in the future be updated to allow for much more flexibility.

5.1 Bar code reader

Reading the bar code was a bit more challenging then I had originally expected. From difficulty with the Android camera to picture quality and memory constraints, the time required to implement the UPC-A reading resulted in it being the only type of bar code the application currently reads.

5.1.1 Difficulty with Android camera

When using the camera of an Android phone there are a few things that need to be remembered. The first is that the camera must be retrieved and released correctly. If this is not done there is a possibility that the phone will need to be rebooted in order to access the camera again. Other applications that use the camera will not be able to gain access to it until the reboot occurs.

Prior to using the camera the developer needs to start the preview. This can only be done once the camera has been retrieved. When releasing the camera make sure to stop the preview before releasing it. This will cause an exception that may result in the camera not being usable by other applications as well as your own.

When setting the cameras properties make sure to perform checks against the hardware to ensure the features that are needed are supported. This includes flash modes, picture sizes, picture types and focus modes. If you specify a picture size that is too large to store in memory you will get out of memory exceptions.

5.1.2 Picture Quality

The quality of the picture being taken is directly impacted by the way the picture is acquired. There are a few ways to acquire a picture once you have started the preview. The first is to fire a take picture event. This will take a picture based on the parameters the camera was configured with. The results that were seen included pictures that were out of focus and very blurry as well as inconsistent flash modes resulting in flares and bright patches.

The quality of the image being take is also dependent on the camera properties specified. The default quality is JPEG which may end up getting compressed leading to a poorer quality image than originally expected. This is a configurable item but could result in out of memory errors if the image size is too large and the type of picture being taken requires too much space.

Taking the next preview frame is the other way of acquiring an image from the camera. The image being acquired usually is a lot clearer and is exactly what is seen on the screen of the phone. The default format for a preview image is NC21, YCrCb which is a color space used in video frames. This format is very easy to convert to a gray scale image. The first width by height pixels are the Y plane information for the picture. The Y plane consists of a gray scale version of the image itself. The rest of the pixel data contains the RGB color information which when applied to the Y plane information results in the color version of the image. For reading bar codes the color information is not needed so the application only uses the Y plane information.

Using the preview frame also allows for continuous auto focusing via internal handlers between the phone and camera. The flash mode used, torch mode, allowed for a consistent light source that would not change after taking the image and allowed the auto focus to correctly focus on the bar code image itself.

5.1.3 Lighting

There are a few different types of flash modes that can be used when acquiring an image from the camera. The modes explored were macro and torch. A macro flash mode is used for close up focus modes. This mode sounds perfect but doesn't actually fire if you are taking a preview image. It only fires when taking a picture which was not used in the final version of this project.

Torch mode turns the flash on to give a constant light source for the picture being taken. When auto focus is called the phone automatically adjust the amount of light being used to get a better quality image of the subject matter at hand. This mode does cause a bright spot on the image but as discussed later the entire image is not used when processing the bar code.

5.1.4 Memory

Android phones have a memory constraint on how much ram an application can use and it is determined by what generation of phone they are. The first generation of android phones allow for up to 16mb of memory per application. The second generation phones allow for 24mb of memory. This information needs to be kept in mind when developing an application that deals with images, especially when those images could be large due to the type of camera and size of image being taken.

For instance an eight megapixel image can be of size 3264 by 2448 pixels for a total of 7,990,272 pixels within the image. If the image is stored in a type that uses one byte per pixel this results in a 7mb image being stored. This is 7mb that the application needs available in ram. If you wish to store the image in higher quality then you need even more storage for the image. For instance if the quality of image is set to ARGB8888, which is an image format that is stored as four bytes per-pixel, then the resulting image is 28mb which exceeds the amount of memory allotted for an application.

Database storage is determined by how much space is available on the internal memory of the phone. This memory can usually be expanded via an SD card and if the application supports the use of the SD card. For this project the database was stored on the phone itself. The amount of storage per row of data is in the bytes range with around 25 or so records taking up 12kb of space.

5.1.5 Types of Thresholding

In order to read the bar code the image acquired needed to be converted to a binary image. This was done by using thresholding methods to convert the image from gray scale to having two possible pixel values, white and black. A few different methods were tried with varying results.

The first method used was an implementation of Otsu’s method. This method worked decent on the whole image but due to bright patches on the image it did not result in a usable binary image for bar code reading. When used on a single line in the image it did not work as well due to there not being enough variations in the classes of pixels the method found. This resulted in inconsistent bar widths and bars missing completely in sections.

Next an iterative approach was used. This approach is the simplest form of image segmentation and involved the continuous averaging of values until the threshold value returned was the same as the previous run. First an initial threshold value was chosen. Being that we needed a value that would distinguish between black and white pixels a value of 190 (out of a possible 255) was chosen. The algorithm would then obtain an average of the pixels that were above the threshold and an average of the pixels below the threshold. The new threshold value was determined by summing these two averages and then dividing the result by two. This method worked well if the image didn’t have too many bright spots and always resulted in a local minimum. If the initial threshold value was changed and used against the same data the resulting threshold may end up different.

Using a modified version of Otsu’s method the application was able to obtain a workable set of pixels to decode. The modified version first took a histogram of the row we are interested in. It then found the largest peak in the data. The algorithm then searched for the second largest peak that was far away from the first. This was accomplished by giving the peak a score based on how far it was from the first peak. The score was calculated by taking the value of the peak and multiplying it by double its distance from the first peak. Based on this score the second peak may not actually be the second largest but it is large enough and far enough from the first to calculate a valid threshold value. The final step of the algorithm is to find the lowest value between the two values. The closer to the center point of the peaks the better and a score is given based on the distance between the peaks. By using the threshold found by this algorithm and applying a very simple sharpen kernel (-1 4 -1) to the row data we are interested in, a valid binary result was found.

5.2 User Interface

Android user interfaces are defined within xml files called layouts. These layouts put together different containers called views that result in different displays of information. This concept is very similar to Java’s Swing framework. Each view is just a container for other views which can result in fairly complex designs. Programmatically creating these layouts is possible but it is much more difficult than using the XML layout files. Some of the attributes that can be set via an XML attribute can take up to three method calls and two object instantiations to get the same result programmatically.

Initially the application used the view called a TableLayout for the display of movie lists, category lists and movie information. The view allows easy addition of new items but does not allow for dynamically updating the display after those items are added. This was an issue when adding a new movie or category to the lists. The user would have to switch to a different display and then back to see the changes take place to the TableLayout.

Not being able to update the display dynamically led to the switching from TableLayouts to ListViews. A ListView is a view that allows items to be displayed in a vertical list but also allows dynamic updates due to its design. The list that is managed by the ListView is stored in an underlying data adapter that fires dynamic change events to the user interface when an item is added, deleted and modified. There were some cases when the changes were not being fired. This was a result of the notify call not happening on the user interface thread. By forcing the call to the user interface thread all the dynamic changes were able to occur without issue. This is the same as with Java Swing events and notifications.

5.3 Database

The Android SDK includes the SQLite database. This is a flat file database that does not use a server for connections like MySQL, PostgreSQL or Oracle. The developer is open to create the database file wherever they choose as long as their application has access to that location. This allows databases to be added to an external SD card if more space is necessary. For this project the database was created on the internal hard drive of the phone as it does not need a large amount of space.

To use the database the developer must first get a database connection. These connections are managed by the Android SDK so if there is already an open database version of the database the developer would like to use it is returned. Internally you can specify which type of database connection you would like, read only, writable, etc. After looking into how these different types of connections are accessed the project always gets a writable version of the database even if we are only reading information out. The Android SDK will return the same database connection no matter which type of database is specified so explicitly stating we would like a read only version wasn’t necessary.

5.4 Web Services

After reading the bar code and validating the check digit the application then calls external web services to acquire the information about the movie we would like to store. If there is no current connection to the internet an error message is displayed to the user. Originally the plan was to use Google web services to retrieve information and then use Amazon web services as a backup in the event the information could not be retrieved from Google. The end result did not use either.

5.4.1 Web Services – Looked into but not used

The Google web services that searched via UPC only returned shopping related results with very few ways to filter the returns. Due to the results being shopping related many of them were from eBay. These were less than ideal as there could be misspellings and/or wrong information about the product within the returned results due to eBay sellers typing in the information incorrectly. The Google web services also only allow for 2,500 calls per day for free. After this quota the developer whose API key is being used is then billed around $5.00 per 1000 additional queries.

The Amazon web services had more consistent results but did not include some of the older products that are no longer sold; i.e. VHS tapes and movies no longer published. Also the Amazon terms of service specify that their services should only be used if there is a plan to bring business to Amazon. The wish list feature of the application could provide a link to search results or product pages but currently does not so rather than violate the TOS, Amazon web services were not used. Also Amazon web services qualify for free usage for up to a year after requesting an API key. The free usage period only supports a certain amount of service calls per day and after a year each service call cost a few pennies. This cost could add up if a user wants to add a large movie library to the application.

5.4.1 Web Services – Used

The application uses two different service providers to gather all of the information about the movie being scanned. The first service provider is the UPCDatabase (www.UPCDatabase.com) which currently allows noncommercial usage of their RPC web services to gather basic information about a product via its UPC. This site does not have information about all possible products available so there is a possibility that a scanned movie may not be found. It does contain a great number of products so the chance of getting a movie that is not stored is very low. It allows up to 20 service calls a day with additional service calls costing $0.002 cents per call that is successful. The service calls use XML-RPC to transfer information between the server and clients. XML-RPC is a remote procedure call protocol that returns results in XML format over HTTP. This is the reason for the only third party API mentioned earlier. Implementing an XML-RPC client for Android would have added a lot of extra time to the project that just wasn't possible.

After acquiring basic information about the movie the application then contacts the Rotten Tomatoes (www.rottentomatoes.com) website to obtain the rest of the information. The Rotten Tomatoes API allows for many kinds of searches but the one we are interested in is the title search. Calling this web service returns a JSON object of the movies that match the title. From this object we can gather the rest of the information about the movie. There are a few things to note about these web services. Rotten Tomatoes does not contain information about box sets, trilogy collections, TV series, etc. It also does not always have all the information we would like for the movies. If the information we need cannot be found then it will have a default value specified by the system.

6. Conclusion

Overall this has been a challenging and exciting project for me to tackle. In the few short months of working on it I have learned a great deal about image processing, memory constraints and Android development. With the Android OS becoming more popular and even merging with the Linux kernel the knowledge I have gained will help with future projects. The project and all of its related files can be downloaded from GitHub via the following URL: https://github.com/pintodragon/CS598MovieCatalog/downloads.

7. References

BarcodeIsland (2006). *UPC-A Symbology* [Online]. Available: http://www.barcodeisland.com/upca.phtml

XMLRPCClient (2012). *android-xmlrpc* [Online]. Available: http://code.google.com/p/android-xmlrpc/

Adams Communications (2012, Jan 19). *All About UPC Barcode & EAN Barcode* [Online]. Available: http://www.adams1.com/upccode.html