Mobile Dev 6.2C

Daniel Coady (102084174)

23/10/2019

Introduction

In this short report, we will discuss how the movie rating Android application was modified to improve on both the performance and the usability of the application. This will mostly cover how UX design was incorporated into the application as well as different technical aspects to keep in mind while developing an Android application.

Performance Optimisation

When optimising for performance there are two main things to keep in mind:

- CPU time
- Memory usage

Everything that you do while developing anything is a tradeoff between space and time. You could either do a calculation one time and store it for later use, or you could do a calculation each time that you need that result. One is not necessarily better than the other and each has their own pros and cons, but it's something to keep in mind while developing any solution. In this case we had an issue of high memory usage, capping out at roughly 75MB of RAM being used at any one time. The reason for this is due to the list view being used in the main activity. A list view by design will create all of the fragments it contains and keep them loaded into memory the entire time, which as you could imagine with a larger list can create some issues. The way to avoid this in this situation was to use a recycler view. Recyler views are similar to list views, with the main difference being that it destroys any fragments that are not in view currently, saving memory. Changing the list view to a recycler view cut down our memory usage by roughly 25-30MB which is a sizable improvement.

Usability

A core issue present in this application is the pause when everything is being loaded in. While this is happening the user is unaware of what the application is doing which makes for a poor user experience. To assist with this, while the file is being read from the application storage we display a progress dialog that informs the user that there is data loading occuring in the background. This means that the user now knows what it happening, giving them peace of mind that the application hasn't become unresponsive.

```
internal inner class loadMovies : AsyncTask<Void, Void, ArrayList<Movie>>() {

    lateinit var loadingDialog: ProgressDialog

    override fun onPreExecute() {
        super.onPreExecute()
        loadingDialog = ProgressDialog.show(this@MainActivity, "", "Loading...")
    }

    override fun doInBackground(vararg p0: Void?): ArrayList<Movie> {
        Thread.sleep(3000)
        return Movie.loadFromFile(resources.openRawResource(R.raw.ratings))
    }

    override fun onPostExecute(result: ArrayList<Movie>?) {
        super.onPostExecute(result)
        loadingDialog.dismiss()

        if (result != null)
            initializeUI(result)
        else
            initializeUI(ArrayList())
    }
}
```

Figure 1: The async task class used to load the data in the background