Performance Optimisation Report

• Introduction

(What does this report contain? What are the key design principles that you will discuss in this report)

This report talks about performance optimisations of lists in android. The two main elements that are used to optimise speed are Async Tasks and View Recycling. Async tasks allow for parallel operation for downloading images while still being able to view the list as they are loaded in. View Recycling is the method of caching resources used and repopulating them with information instead of deleting and recreating the layout each time you scroll.

• Performance Optimisation

(What was the problem as observed? What is the root cause of the problem? What was the design approach used to correct the problem? Code snippets can be used to illustrate the problem/solution)

The main problem that caused the stutter and lag was when the user scrolled through the list. This was due to creating, loading, looking up and deleting the elements as the user scrolls. To negate this issue you can recycle the elements from the previous parts of the list. This process is shown below.

This code caches the layout objects so that they don't need to constantly be deleted and recreated and relocated each time you scroll, instead once they are created they can be reused over and over.

```
//view lookup cache
private class ViewHolder
{
    ImageView icon;
    TextView movieText;
    TextView votesText;
}
```

This code displays the main part of the optimization with reusing views once they are created. The code within the if statement is only ever run when the views have not been created at the start of the app running, which it then creates the views and caches the viewholder values for the layout so it can reuse them.

```
public View getView(int pos, View convertView, ViewGroup parent)
{
    //get data item for this position
    //Movie currMovie = getItem(pos);
    Movie currMovie = movies.get(pos);
    ViewHolder viewHolder;
    //check if an existing view is being reused, otherwise inflate the view
    if(convertView == null)
    {
        //if there is no view to reuse, inflate a brand new view for row
        viewHolder = new ViewHolder();
        //LayoutInflater inflater = LayoutInflater.from(getContext());
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.listrow, parent, false);
        //looku view for data population
        viewHolder.icon = (ImageView) convertView.findViewById(R.id.row_icon);
        viewHolder.movieText = (TextView) convertView.findViewById(R.id.row_label);
        viewHolder.votesText = (TextView) convertView.findViewById(R.id.row_subtext);
        //cache viewholder object inside fresh view
        convertView.setTag(viewHolder);
        Log.w("creating", " new view");
    }
```

The else statement is part of reusing the views if they had already been created. The if statement includes assignment of the values within the view with the reused layout through the viewholder cached values as shown with viewHolder.

```
else
{
    viewHolder = (ViewHolder) convertView.getTag();
}
//populating the data into the view
if (currMovie != null)
{
    viewHolder.movieText.setText(currMovie.getName());
    String votesStr = currMovie.getVotes()+" votes";
    viewHolder.votesText.setText(votesStr);
    Bitmap movieIcon = getMovieIcon(currMovie.getName(), currMovie.getRating());
    viewHolder.icon.setImageBitmap(movieIcon);
    Log.w("MMMMMMMMM", "Creating row view at position "+pos+" movie "+currMovie.getName());
}
```

• Usability Improvement

(What was the problem as observed? What was the cause of the problem? How was the problem corrected? Code snippets can be used to illustrate the problem/solution).

For future use the Async task with progress dialog will be useful for when loading pictures and larger amounts of content. In this case all of the data was loading too fast for the progress dialog to appear for very long. In cases with large pictures or downloading data from a server using Async tasks with progress dialog or just in the background would work really well.

Here is the code displaying the progress dialog window that pops up before the Async task begins its background work.

```
protected void onPreExecute(){
    super.onPreExecute();
    pd = new ProgressDialog(MovieRatingsActivity.this);
    pd.setTitle("Processing...");
    pd.setMessage("Please wait.");
    //pd.setCancelable(false);
    //pd.setIndeterminate(true);
    pd.show();
}
```

This is the part of the Async task which is doing all the work, loading the information from file.

```
protected ArrayList<Movie> doInBackground(Void... v)
{
    ArrayList<Movie> m = null;
    try {
        InputStream inputStream = getResources().openRawResource(   R.raw.ratings);
        Thread.sleep(5000);
        m = Movie.loadFromFile(inputStream);
        inputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return m;
}
```

Here is the Async task concluding with closing of the Progress dialog and setting the information to the list.

```
protected void onPostExecute(ArrayList<Movie> m) {
    super.onPostExecute(m);
    movies = m;
    setListAdapter(new RowIconAdapter(con, R.layout.listrow, R.id.row_label, movies));
    if(pd != null)
        pd.dismiss();
}
```

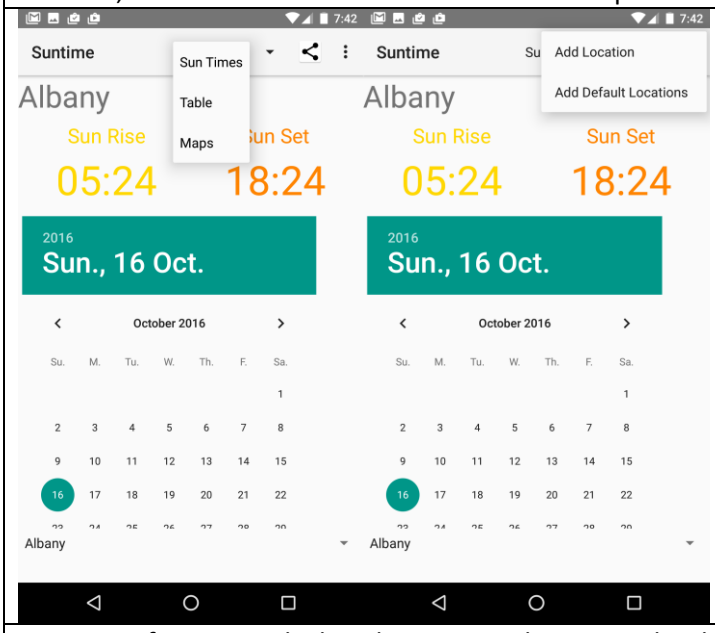
• References

• Appendix (Commented code snippets as needed to support the report)

Task 2

(a) list out the fragments in your application

The action bar includes the tabs in a drop down list to swap fragments also includes commands such as share, add location and add default location to perform the actions listed.



Sun Times fragment which is the same as the activity but has been converted to a fragment.

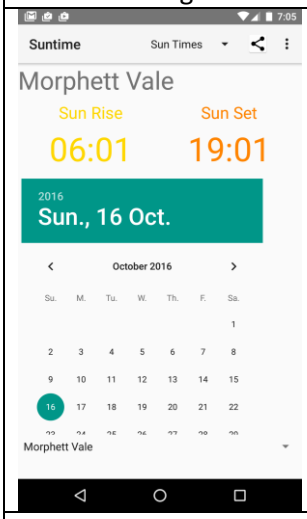
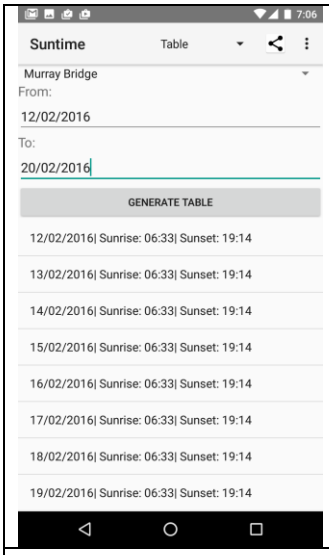
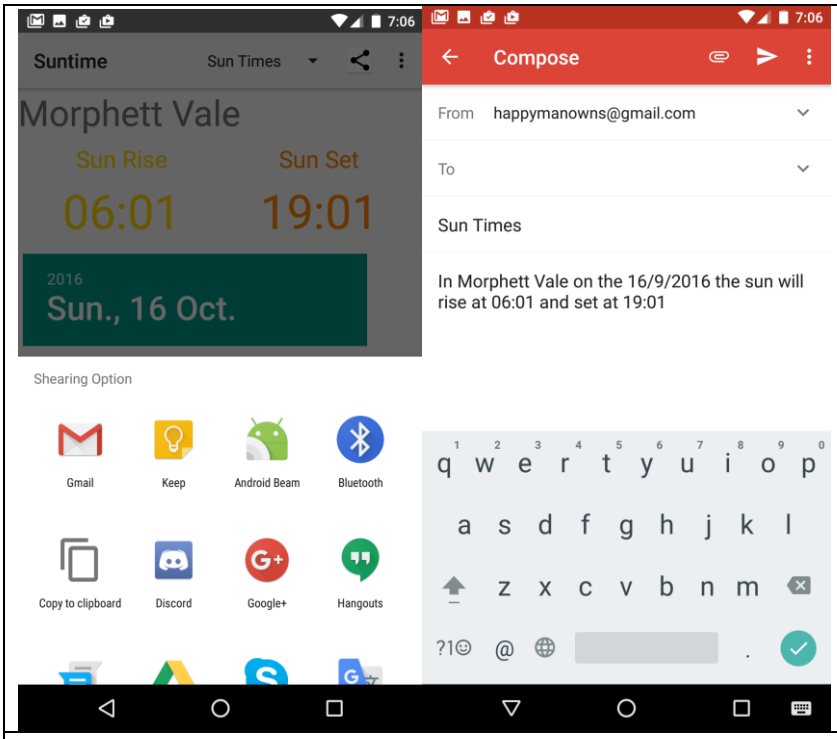


Table fragment which allows you to select a suburb and then select a date range in which you generate a table that includes all the sun rise and set times for every day in between the dates posted.

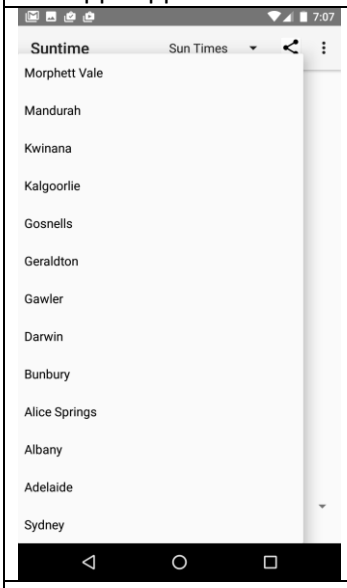


Maps fragment that lists the location and sun times with integration with google maps so the user can see where and what the times are for a specific location from other tabs.

Share feature that only works when you are on the sun times fragment and will allow you to share the location, date and sun times over an intent. Gives you a toast if not on the sun times fragment



The app supports all the major Australian Locations



The fragments were swapped in a spinner element which first checked if they already exist then if not created a new instance of the fragment and replaced the currently displayed fragment

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{
    // On selecting a spinner item
    String item = parent.getItemAtPosition(position).toString();

    switch(item)
    {
        case "Sun Times":
            replaceFragment(frag);
            break;
        case "Table":
            if(st == null)
                st = new SunTable();
            replaceFragment(st);
            break;
        case "Maps":
            if(mf == null)
                mf = new MapFragment();
            replaceFragment(mf);
            mf.getMapAsync(this);
            break;
```

Here is the code that actually replaces the current fragment with whatever fragment is added to the parameters.

```
void replaceFragment(Fragment f)
{
    android.app.FragmentManager fragmang = getFragmentManager();
    android.app.FragmentTransaction transaction = fragmang.beginTransaction();
    transaction.replace(R.id.fragment_container, f);
    transaction.addToBackStack(null);
    transaction.commit();
}
```