CALIFORNIA STATE UNIVERSITY
LOS ANGELES

Los Angeles, California

Lunar Exploration Web and Mobile Applications

Lunar Mapping and Modeling Portal Android Application

(LMMPAA)

CS496 Senior Design

Prepared by:

Eddie Arevalo
Alvaro Ortiz
Daniel Soto

Table of Contents	
Title	Section

1.0 EXECUTIVE SUMMARY:

The Lunar Mapping and Modeling Portal is a website that allows access to and browsing of lunar data collected by JPL. This information is primarily but not limited to elevation maps of the lunar surface. These maps were collected by different lunar missions such as the LRO and Clementine with a variety of instrument types. Further data consists of a more mathematical or historical nature such as sun lighting across the moon, distances or the historical reason of how a crater came to be named.

The LMMP website already had most of this data featured in an intuitive way. When the first meeting arrived we learned what the patron wanted was 3 different projects that coincided closely to the website. The first was a reimplementation of a part of the website that would generate images depending on the light sources on the moon. This was called the lighting tool(LMMPLT). The second was an android application version of the LMMP website. The name for this application was Moon Tours although we referred to it as the LMMP Android Application(LMMPAA). The third was an application either web or local that would allow the user to see 3D real time images of what someone standing on the moon would see. We called this project the Virtual Photograph Generator(VPG).

As time went on there were many changes to what the requirements for these projects should be. the LMMPLT and the LMMPAA consumed most of the time that would have been used on working with the VPG. The postponing of the VPG eventually led to it being dropped as a requirement as it become more of a priority for the other two projects to be completed rather than have three uncompleted projects. In this way the LMMPLT and

the LMMPAA were completed on time. These two projects had working versions although they were refined to a point that was acceptable for release.

2.0 INTRODUCTION:

The purpose of this document is to give a good understanding in the use and architecture of the LMMPAA. The LMMPLT will be explained in a separate document. This document will be beneficial to anyone seeking to change or maintain the LMMPAA's code. The information for this can be found in three sections. The first is in secton 3.0 User Guide which will foster understanding in the intended use of the LMMPAA. Secondly for an explanation of the architecture and design in which the LMMPAA was implemented the reader should refer to section 4.0 Architecture. Lastly a reflection upon the lessons learned by presenting which approaches worked and which did not the reader should refer to section 5.0 Conclusions. The conclusions section will point out several flawed ideas that we encountered during the building of this software. It would be especially helpful to anyone working on this project to learn from our mistakes. As Otto Von Bismarck once put it "A smart man learns from his mistakes, but a truly wise man learns from the mistakes of others," or as one of our own CSULA professors put it "It is expensive to learn from your own mistakes, but it is free to learn from the mistakes of others."

3.0 USER GUIDE:

The LMMPAA primary purpose is to show maps of the lunar surface. As such when first opening the application the user will first be brought to the map screen. Here the user will be able to pan across the lunar surface map. From this point there are 7 options in the menu bar. The options are Search, Bookmarks, Layers, Nomenclature, Markers, Minimap and Settings.

3.1 Search:

The search option allows for a user to type or voice a keyword. This will then show results for that keyword in a list by nomenclature, layer or bookmark. By selecting the nomenclature the user will be shown that location on the map. By selecting layer the user will be directed to the details of that layer. By selecting bookmark the user will taken to the location of that bookmark on the map.

3.2 Bookmark:

The Bookmark option takes the user to see a list of the bookmarks brought by the application as well as any bookmarks the user might have added themselves. By selecting a bookmark from this list a more detailed view will appear. This view contains information such as the history, an image and a button to show the location on the map. The user is also able to add their own bookmarks by selecting the add bookmark button.

3.3 Layers:

The Layers option takes the user to a menu screen which allows the user to select which layers, base maps and special options they want to see on the map. The user can add layers by pressing the green plus symbol at the top of the layers section. This will take them to a list which can be sorted by product type, mission, nomenclature and instrument type. By selecting these layers they are added to the map. The base map can be changed by selecting from the drop down list. Special options can be toggled by checking the box next to their name.

3.4 Marker tool:

The Marker tool allows a user to draw out shapes on the lunar surface. Data over these shapes can be obtained by selecting them. These shapes can be drawn by point, line, rectangle, circle, polyline, polygon or freehand. When the user activates this tool all of these options will be immediately selectable.

3.5 Minimap:

The Minimap option allows for the toggling of the minimap. This minimap shows the users relative location of the screen over the map. This allows the user to know at what position they are even when they are zoomed in. The minimap can also be used to pan across the lunar maps.

3.6 Settings:

The settings option allows the user the change several preferences that they would like the application to use. This allows the user to set cache size or marker colors.

4.0 ARCHITECTURE:

The LMMPAA is composed of 9 modules. These modules consist of the 7 menu options plus the map view and the utilities. The application is based on an MVC pattern where the data is kept static, the user interface shows the data and the controllers within those user interfaces allow manipulation of the data. All the data comes from the JPL REST services although the larger portion of this data is kept within the application assets folder in order to greatly increase load time during operations. This is done because the load time to load directly from the rest server is not fast enough to keep up with the demand of any enjoyable use. The packages in the code are organized by MVC conventions. The UI is separate from the model and it is separated from the static data. The application can be explained better by the visualization of modules. The way these modules work together is how this architecture will be explained.

The most important aspect of the Moon Tours application is to be able to browse the moon maps. This is the reason that the first thing that the user will see when opening this application is a screen which allows for movement across the map. This map is the main activity of the project as well as the place where all other activities will either manipulate or somehow point to a location on the map. This activity is powered by the ESRI Maps API. The graphics, layer order, opacity and map callouts are handled by this API.

Before the map view can be shown all the data from the xml and json files from the JPL REST service must be extracted by using the utils portion of code. These classes are inside the utils folder. They are called by the map view on startup via an async task. They

handle the reading and placement of the data into the model classes. The async task was used in order to allow the user to browse the map while the nomenclature and layers data is parsed from the XML files in the background.

4.1 Singletons:

The filled model classes are held within Data singletons. These singletons function as the data base from which data can be sent to the user interfaces. These can be split into 2 categories. The first consists of the XML and JSON file information. These include the basemaps, config, layer, nomenclature, tile and tiled group Singletons inside the data folder. The singletons support getting information but not writing as this information is static and must not be changed in order to continue allowing the application to work correctly. The second type allows writing as these classes are meant to keep track of the user selected manipulations of the map view. This data can be changed at any time depending on what the user selects. These include the bookmark, mapdata and markerdata singletons.

4.2 Modules Introduction:

The Search, Bookmarks, Layers, Markers, Minimap and Settings modules make up the 7 other modules. These modules manipulate or show details about the main map view. Each one is connected to its own Singleton class which stores their currently selected data. Each one makes use of Android classes such as lists, fragments, activities and gestures. These function as the view and controller in the MVC pattern.

4.3 Search:

4.4 Bookmarks:

4.5 Layers:

4.6 Nomenclature:

4.7 Marker tools:

4.8 Minimap:

4.9 Settings:

5.0 CONCLUSIONS:

5.1: Reusing software:

Many lessons were learned from this project but the most important one is about refactoring. Originally an application had been given to us to work with. We attempted to use this code as a base to work off of but this turned into a mess as the previous and new bugs mixed together become almost impossible to fix. The existing bugs were the most difficult to find as we had no idea where they were coming from. Having spent the entire first quarter bogged down we realized that we would have to rewrite the entire application. After the application core was rewritten we added all the modules one by one while testing each one individually. If we had done that from the beginning we would have finished the project much faster. This is one important lesson that we have learned. If the code is already buggy do not add to it. It is time to refactor or remake it.

5.2 Proper use of time:

A second lesson that was learned is how much can be accomplished within a limited set of time while attempting to handle the other courses. As this was not a normal class a specific time to work was not set up and instead work was done as time was available. Usually in the fashion of doing the more immediate course assignments before working on the senior design project. In order to combat this lack of sense in order. We set up goals to accomplish by the week in order to know if we were on track or not. The original quarterly plan quickly fell apart and we were forced to plan by the week although most of the

major goals could be moved to fit a better schedule. Having deadlines on this smaller scale and having them due to the professor during meetings pushed us to work harder and have something working in order to show it during that meeting.

5.3 Android programming:

The third lesson was programming in Android. There was a learning curve in that none of the team had ever worked on Android applications before. This meant that every move by anyone on the team would have to be thoroughly researched first by use of documentation, examples and tutorials. It was quite often that a plan for implementation failed due to a previously unknown part of the plan being unable to be done the way it was planned. Such as not being able to use lists to show certain information or not being able to manipulate views because they were referenced and there was no way to track their movements. This weighed heavily in the amount of time consumed as much work was done but by manner of it being research there would be nothing to show for it. We read the books and did tutorials before beginning the project but this was almost useless as most of what was learned was not encountered in this project. We did however notice many Android programming elements that we would commonly run into and wish we had known ahead of time to focus our study on these elements.

The elements of Android programming that we commonly ran into were plenty. A good list for anyone preparing to work on this project is as follows(in no particular order):

- StartActivityForResult
- Custom ArrayAdapter
- Custom ListAdapter
- Custom ExpandableList Adapter
- Singletons

- Parsing Json
- Parsing XML
- Pager
- Fragments
- Callbacks
- Handling different screen sizes
- Search widget
- Touch listeners
- Spinners
- XML styling
- XML widgets
- XML layouts
- LRU cache

The most effective way to use these is to make a working simple version in a different project using similar dummy classes. Write down the steps you took to implement it and then follow those steps to redo it inside the main project with the actual classes. This way confusion is minimized and testing runs are shorter as running the dummy project is much faster than running the main program every few minutes.