CS486C –Senior Capstone Design in Computer Science
Project Description

Project Title: Leaflet 1.5.x for Planetary Mapping	
Sponsor Information: 	**Trent Hare (thare@usgs.gov)** Anderson Moyers (amoyers@contractor.usgs.gov) Scott Akins (sakins@usgs.gov) Astrogeology, United States Geological Survey (USGS)

Project Overview:

The Astrogeology Science Center, part of the United States Geologic Survey, provides support for several NASA satellite and robotic planetary missions (Mars, the Moon, Europa, Titan, etc.). Not only do we support the data processing software for the gathered data, but we also support the infrastructure required to serve the data to scientists, students and the general public. These data sets are generally huge global image mosaics showing the surface of Mars or topographic data showing the elevation heights of the Moon.

In general, we try to adapt existing software instead of building our own software from scratch. However, a persistent challenge in supporting such streaming services is the velocity at which the current available open source mapping applications improve and evolve. Two web mapping applications that continue to outpace all others include OpenLayers and Leaflet. Both are very capable open source projects based on JavaScript. As in other industries, we continue to see JavaScript rapidly replacing JAVA, .NET, and Flex for interactive web applications.

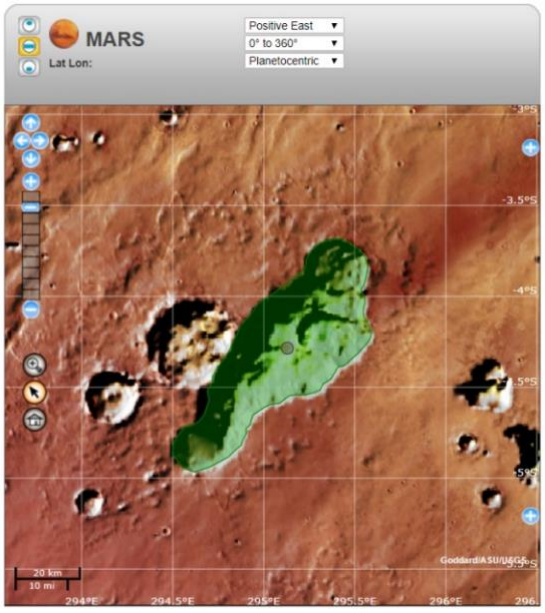

Figure 1. Example viewer allowing specific planetary user options. This map shows one of our viewers using the outdate OpenLayers v2.x. While the Leaflet interface is different, the resulting user options and available graphics (e.g. scalebar) should be similar.

Over the last decade, we have extended a product called OpenLayers 2.x with special customized functions that allow it to manipulate planetary datasets. These extensions are required to properly view and interact with our planetary data sets including supporting custom body sizes (Mars is much smaller than Earth) and the listed coordinates on our planetary maps can differ than what all Earth-based applications currently support. Thus, with these planetary extensions, the OpenLayers mapping application has been extremely effective for allowing our scientists and the general public to explore the diverse planetary data services we

support. This interface is also critical for us to additionally overlay information like internationally-approved named features and their defined boundaries (think of city boundaries or street names on Google Maps).

This year, we are now in the process of updating our planetary extensions for OpenLayers 2.x to function with the current OpenLayers 5.x version. At the same time, we really see the need to also support the other main mapping applications Leaflet. Like OpenLayers, Leaflet is not currently able to handle planetary datasets and will require the same extensions. Leaflet is becoming more popular in data science interactive environments like Jupyter notebooks. Thus, whole we still need to support or current web mapping application using OpenLayers, we see a great opportunity for employing Leaflet also. Our main customers for Leaflet include the planetary researcher who finds the simple OpenLayers environment too rudimentary. Jupyter notebooks have expanded the way scientist interact with data sets (generally through Python). But during a Jupyter notebook session, a simple call to create an interactive Leaflet mapping interface can be instantiated by the scientist to more readily visualize their data. But Leaflet must be upgraded to support the same planetary extensions we plan to upgrade in OpenLayers.

Introduction to Leaflet. Leaflet is a web-based viewing tool which was created for (Earth-based) mapping applications. Here is a brief overview drawn from their website https://leafletjs.com/: *Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 38 KB of JS, it has all the mapping features most developers ever need. Leaflet is designed with simplicity, performance and usability in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.*

Project Details:

As outlined above, the aim of this project is to create a plug-in extension to the Leaflet 1.5.x web-based mapping interface to allow it to support planetary data sets and the functional needs of users needing to explore such datasets. The technical goals for this project include modifying existing Leaflet functions and/or adding new Leaflet function to allow for additional options to support planetary data, as well as more robust support for non-Earth data sets. The language to be used is JavaScript (JS). USGS will provide all the data layers which are available from existing online services, as well as technical support in terms of understanding the basics of planetary mapping and the functions that need to be implemented. Specific features to be supported in the new Leaflet Plugin include:

- Plugin must be supported within all modern web browsers.
- Plugin must be supported within Jupyter Notebooks.
- Variable radius for measures, scale bar display, and used in map projections. To support arbitrary planetary bodies, one needs the mapping application to support custom radius values; most common applications (like Leaflet) assume that everything is happening on Earth (usually

tagged as WGS84) and have this radius hard-wired in. Thus, at the initialization of the mapping interface the user needs the ability to pass in their own radius.

- Able to report East or West Longitudes. Reporting East is the default for Leaflet and Earth maps. Some planetary bodies are also reported in positive West Longitudes (where the Longitudes increase to the left).
- Able to report -180 to 180 or 0 to 360 Longitude ranges. Along with West Longitudes, some planetary bodies are standardized on 0 to 360 Longitudes. This is also a simple adjustment using the math modulus function.
- Support both ocentric and ographic latitudes. Ocentric latitudes will be the default in standard earth-based viewers. Ographic latitudes are an adjustment to the Leaflet reported latitude based on an elliptical body (when the semi-major and semi-minor radius values are different). Leaflet has a method to display grid lines for the major Longitude and Latitudes. This function needs to be adjusted to show East/West, 0 to 360, and ocentric/ographic latitude systems as set by the user.
- Display gazetteer name tagged/symbolized by type. Nearly all bodies we have study have named features on the surface. At USGS, we support a live mapping database where these can be shown per body. These must be able to display correctly within Leaflet.
- Display name polygons and lines from gazetteer. Along with the names, we also support a linear or polygon feature per name within Geoserver. This function must also be supported in Leaflet.
- Other features to make the Leaflet plugin as useful, flexible, and easy to use as possible, as discovered during the requirements acquisition process.

Knowledge, skills, and expertise required for this project:

- Basic knowledge of web application and use of modern web application frameworks, sufficient to understand integration on Leaflet.
- Knowledge of Leaflet, the technologies it relies on, and how to modify/extend Leaflet applications. Available from their website and numerous user forums.
- Basic knowledge of planetary mapping and map-making functions. Support will be provided by USGS professional staff to get the team up to speed.

Equipment Requirements:

- There should be no equipment or software required other than a development platform and software/tools freely available online.
- The team should develop on their own machines/platforms during the development phase; access to necessary USGS computing environments will be provided to support testing and final deployment.

Deliverables:

- The software applications as described above, deployed in a designated USGS environment and tested successfully with real data and real users. Must include a complete and clear User Manual for configuring and operating the software; preference will be for online version attach to the Leaflet extension.

- A strong as-built report detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in GitHub, BitBucket, or some other version control repository; and as a physical archive on a USB drive.
- True success would be a full open-source cycle, i.e., application for review and successful push of this module into the official Leaflet project as a extension.