CS4962 Senior Design

Planetary Surface Flyover Movie Generator

Project Documentation

Document Prepared by:

Shawn Anderson
Angel Jimenez
Khang Lam
Christopher Omlor
Hieu Phan

April 28, 2017

CALIFORNIA STATE UNIVERSITY LOS ANGELES

Los Angeles, California

Table of Contents

1. Executive Summary

The Planetary Surface Flyover Movie Generator project was a first year project with a primary requirement to generate an easy to use system for a user to simulate a flight path over a planetary surface while maintaining scientific accuracy without sacrificing performance and output a high quality video.

The first month and a half of the Fall semester was spent preparing for the project. We first fill in the major roles of the project and specified the meeting times between the group and the liaisons. We clarified the exact requirements that our clients wanted and we then figured out the best technologies and approach to the project would be. We reached an agreement to use Cesium,Spice, and Blender. We did extensive research on each of the technologies to maximize the efficiency. By the end of September we had a general idea of the flow of the project.

In October we started integrating ideas to Front-End, Web Controller, and Back-End. For the Front-End we set up basic functionality for the user interface, decided on using JSON files to send information to backend, and began setting up the rendering. For Web Controller, we set up spice as a web server host to be able to call when retrieving sun data and found a Python wrapper for the NAIF SPICE C toolkit named Spiceypy. In the Back-End we created a JSON parser, render server, and a hadoop cluster to split the work.

Toward the end of the Fall semester all three major parts of the application had working

demos. In the Front-End we had a demo in which the user can add, delete, and edit points, using the points the user created we were able to show a preview in first and third person. Web Controller’s demo was using Spiceypy to calculate the position of the sun in the form of a vector from the center of Mars to the sun. Back-End’s demo consisted of importing obj file, rendering and animating by scripts, and showed a video. Also improved the rendering times of each frame from 3 minutes each to 4 seconds each.

Over Winter break we were able to make the sun data more accurate, instead getting the vector from the center of the planet we get the vector from a point that the user makes. Our main goal for the break was to solve integration of the Front-end and the Back-End. This proved to be more of a challenge than anticipated. By the end of the break we only had an idea on how to solve this problem but were nowhere near a solution.

It was not until about the end of February that a full solution to connecting the Front-End and backend was decided upon and implemented. During that time it was also discovered that the Front-end coordinate system was vastly different than that of blender. This led to discovering more problems relating to the coordinate systems. We found a dataset that covered about 17Km by 52Km of the surface, the dataset was about .5 per pixel accurate and 584 megabytes in size. The dataset we are using was not mapping to the proper location on the globe which was causing the camera system on the Back-End to be put in the wrong location. After developing some mathematical equation to translate the coordinate systems and also fix the improper placement of the dataset on the Front-end we discovered that the camera rotations did not match either. By the end of March we had finally resolved all the inconsistencies in the two systems and were able to generate a model on the Back-End the showed the same movie as the user sees in the preview on the Front-end . At the beginning of April we had a functional software system that would generate a movie directly from the Front-end system and render the movie using hadoop map-reduce. The final part of the application that we completed was to give a link to the user's email indicating where they can download the video.

2. Introduction

The Planetary Surface Flyover Movie Generator project is a system that generates a user-defined flyover movie from the planetary Trek portals (http://marstrek.jpl.nasa.gov, http://vestatrek.jpl.nasa.gov). Application allows users to specify a designated camera path. Each point of camera path contains information for camera position(latitude, longitude, elevation), camera orientation(roll, pitch, yaw), camera viewing angle, and data layer being displayed. Application allows user to interactively add information through a web interface and generates a movie by rendering and animating scenes from the camera's perspective.

The requirements of our project from our Liaisons were to utilize the Trek portals and to render the movie as scientifically accurate as possible. Render times were required to be at approximately one frame per second. Our liaisons also suggested it would be nice if the rendering was done utilizing hadoop mapreduce. Throughout the project additional requirements were added for adding user selectable textures on the front end and having those textures rendered in the final movie.

Due to the complexities of this project as well as the new technologies that were required to learn. We decided on an agile development methodology. This allowed us to learn the new technologies as we progressed in the project and develop modules progressively. We initially split into two groups for frontend and backend. The frontend team researched the cesium software to understand the inner workings of the system to use as our user interface.

The backend team researched various three dimensional modeling programs to find an open source solution that allowed full automation through an application programming interface. We settled on blender for the backend for the fact that it is the only open source solution that meets our requirements.

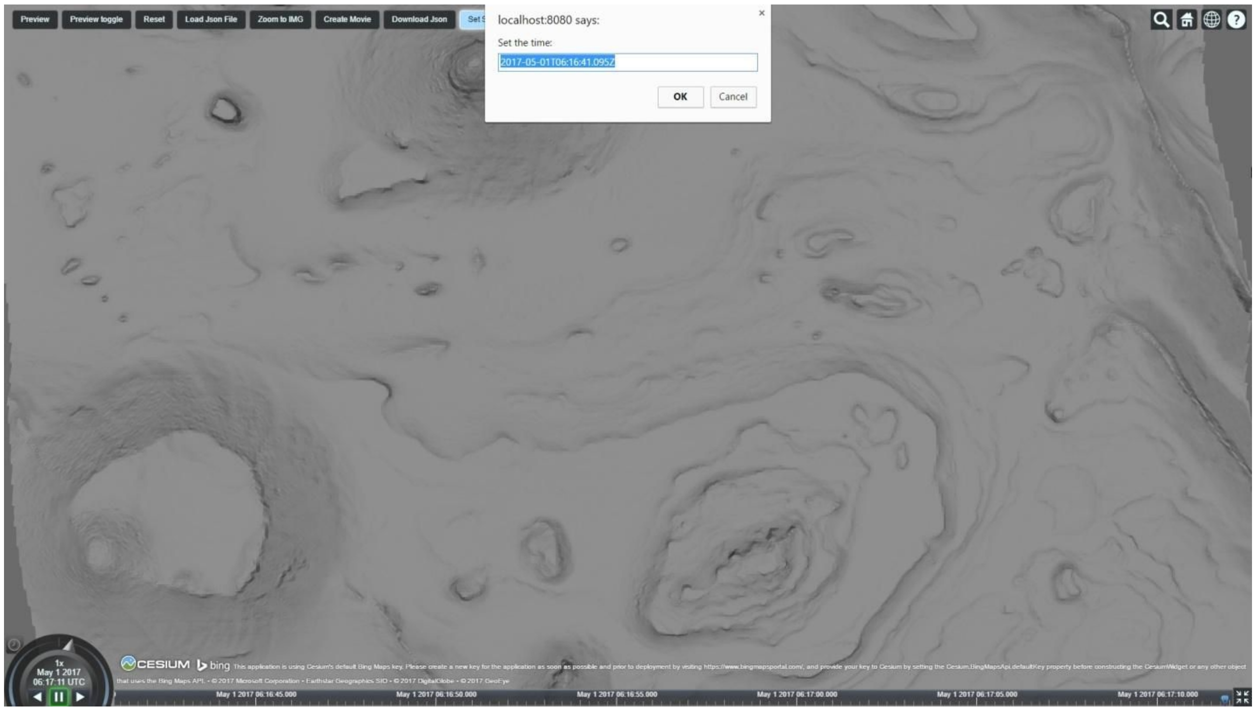
After deciding on the software platforms and getting familiar with the environment we started assigning areas of the project for each team member to work on. Frontend was divided into user interface and preview modules. Backend was divided into sun data, scene creation and automation, as well as rendering optimization.

In December 2016 we were able to individually show a functional user interface as well as demonstrate rendering a movie. At that time the frontend and backend were two separate components and could not communicate with one another. The model was not scientifically accurate and rendering times were terrible, averaging around 9 seconds per frame.

3. <u>User Guide</u>

Set Start Time

- User sets the time to start



localhost:8080 says:

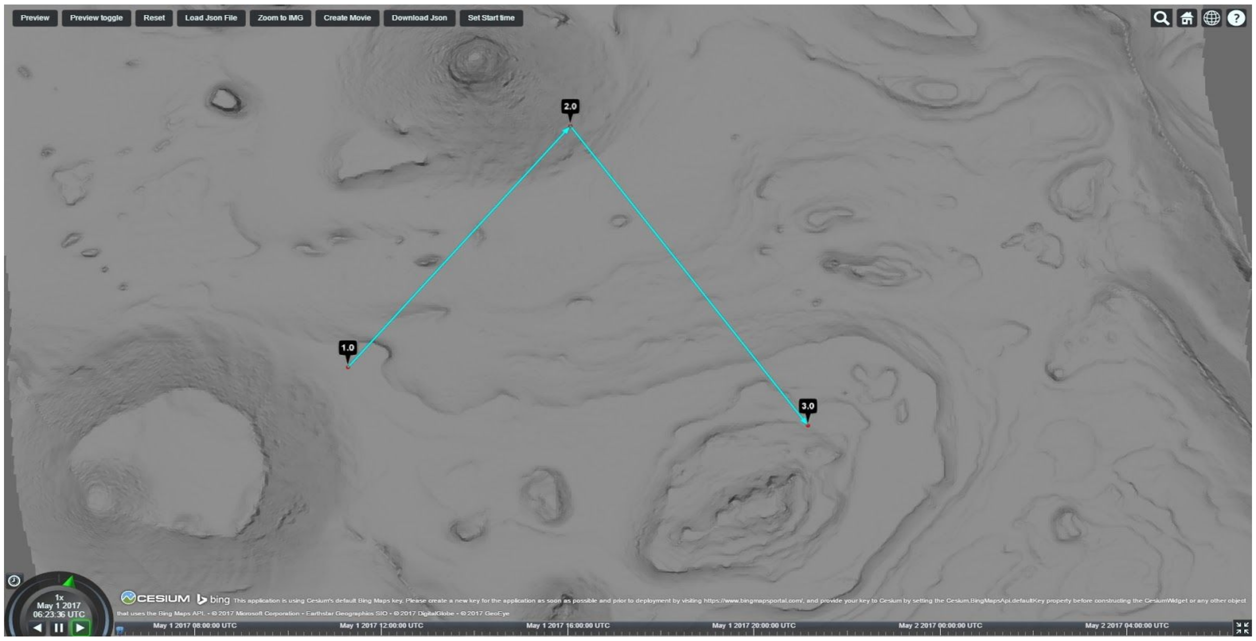
Set the time:

2017-05-01T06:16:41.095Z

Points

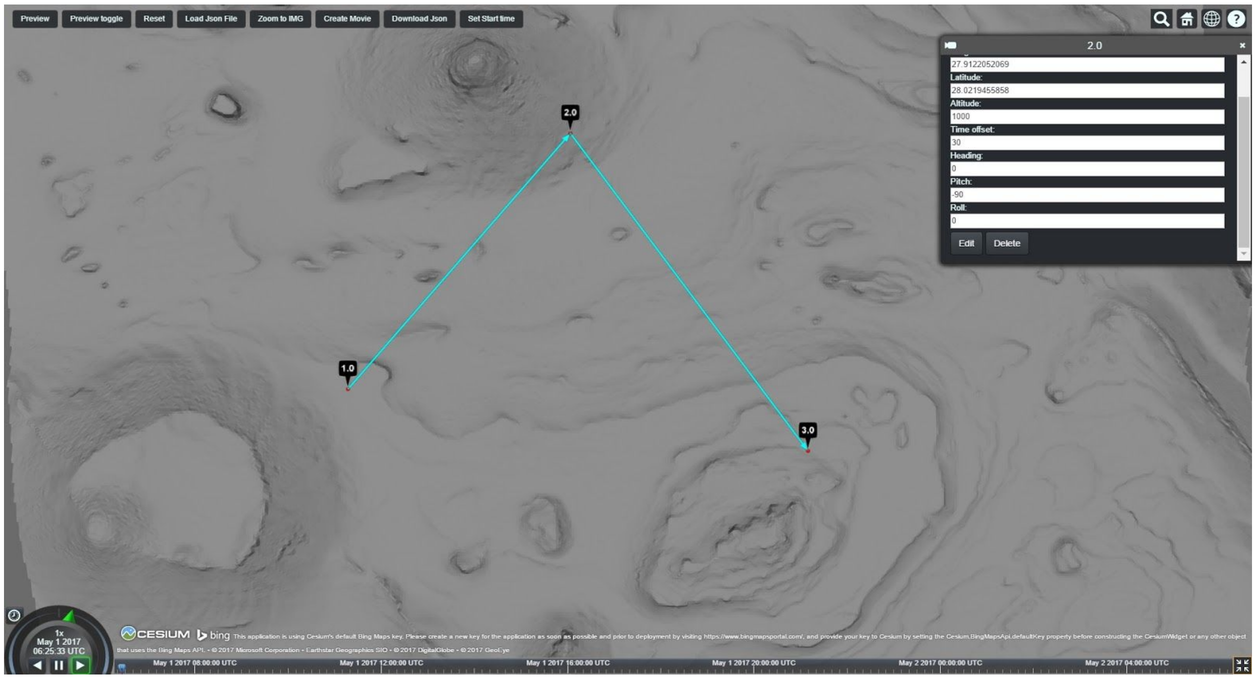
Create Point

- Creates point where user double clicks.
- Automatically creates the path of the user selected points.



Edit

- User right clicks existing point.
- Point information pops up.
- User then right clicks the portion that needs updating.
- Enters change.
- Clicks edit button.



2.0

27.9122052069

Latitude:
28.0219455858

Altitude:
1000

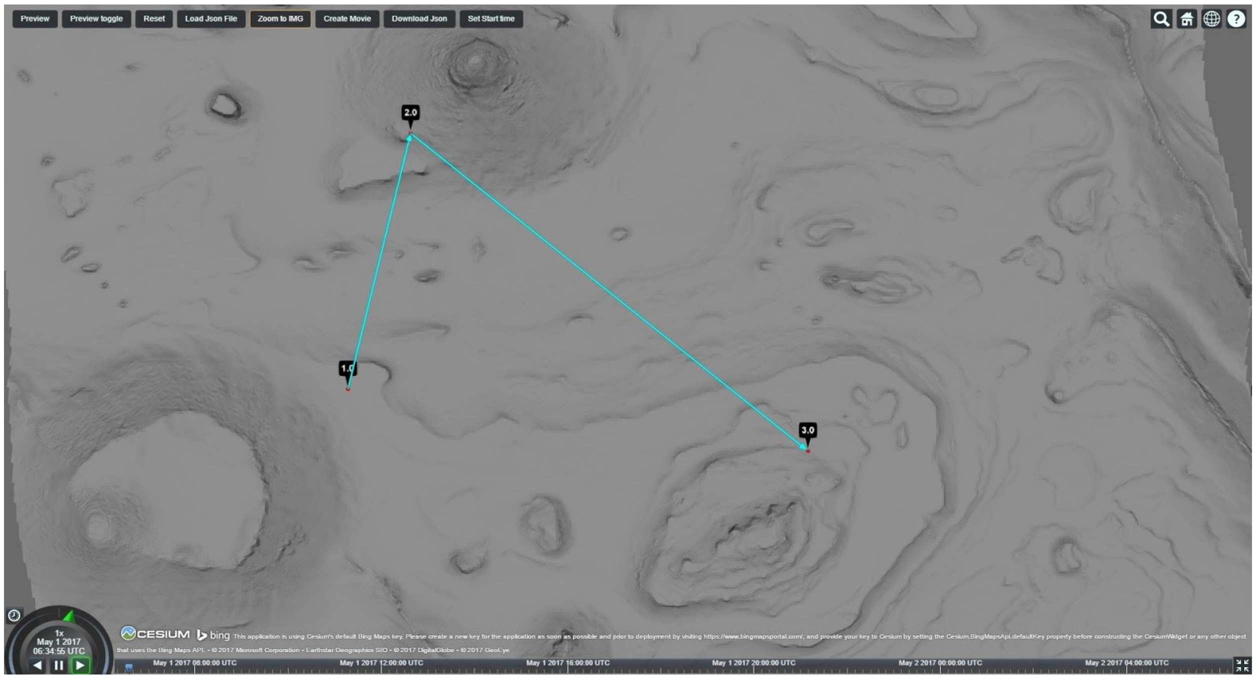
Time offset:
30

Heading:
0

Pitch:
-90

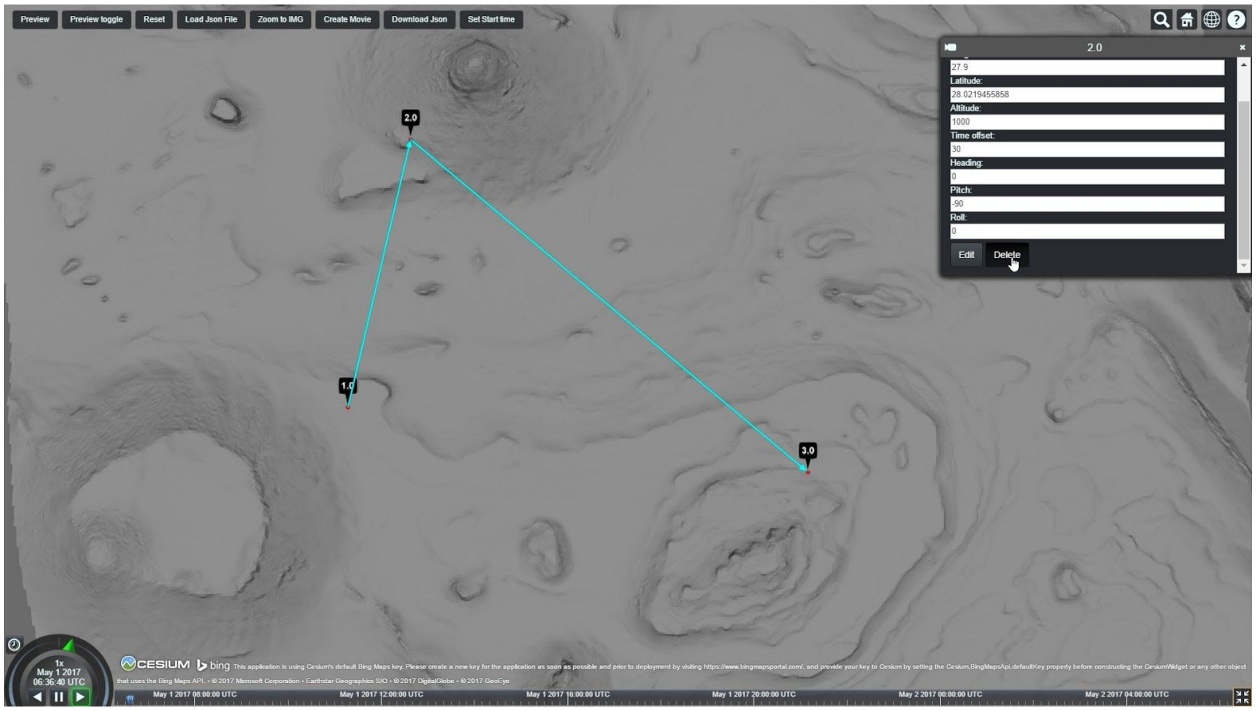
Roll:
0

Edit Delete



Delete

- User right clicks existing point.
- Point information pops up.
- Users then right clicks the delete button.
- Point erases and the path is updated.



2.0

27.9

Latitude: 28.0219455858

Altitude: 1000

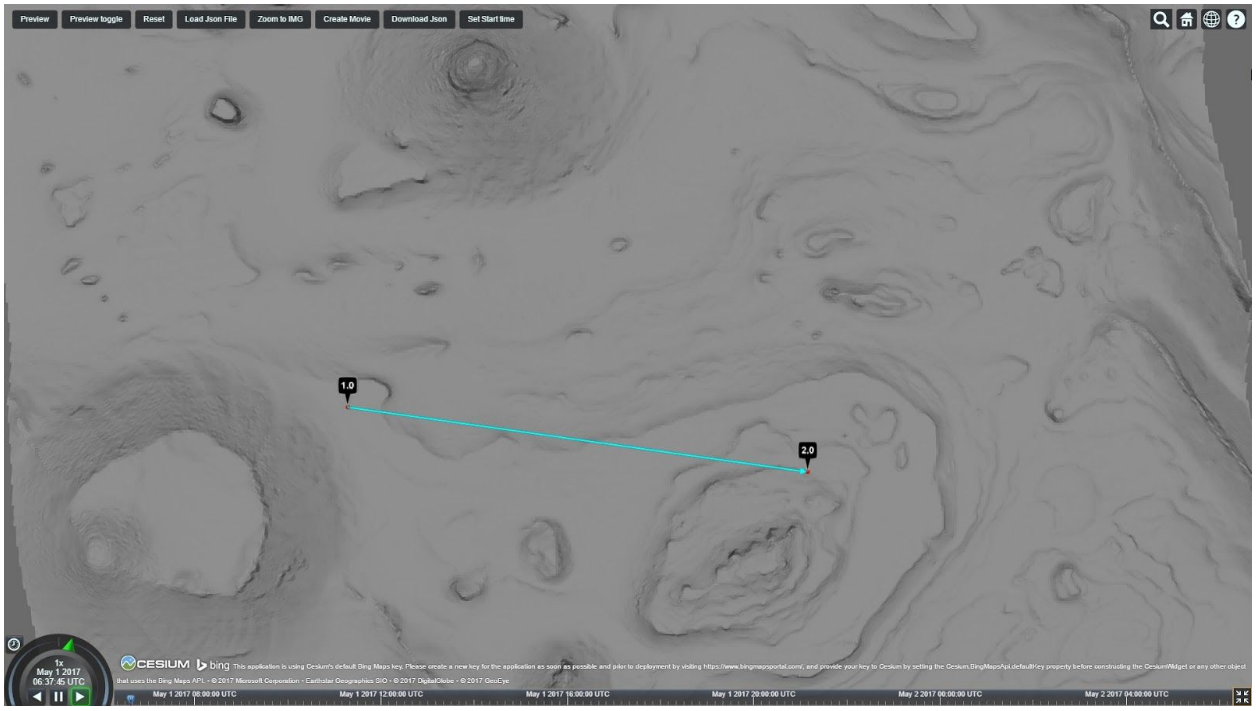
Time offset: 30

Heading: 0

Pitch: -90

Roll: 0

This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • © 2017 Microsoft Corporation • Earthstar Geographics SIO • © 2017 DigitalGlobe • © 2017 GeoEye

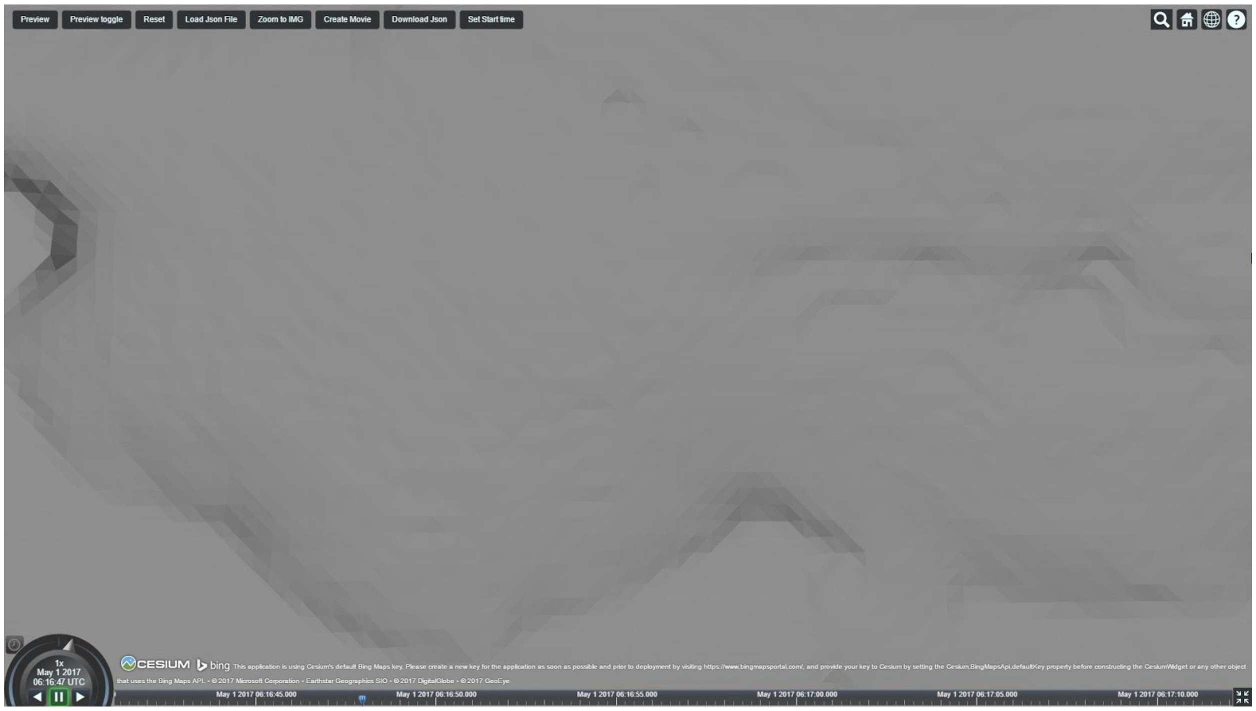


This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • © 2017 Microsoft Corporation • Earthstar Geographics SIO • © 2017 DigitalGlobe • © 2017 GeoEye

Buttons

Preview

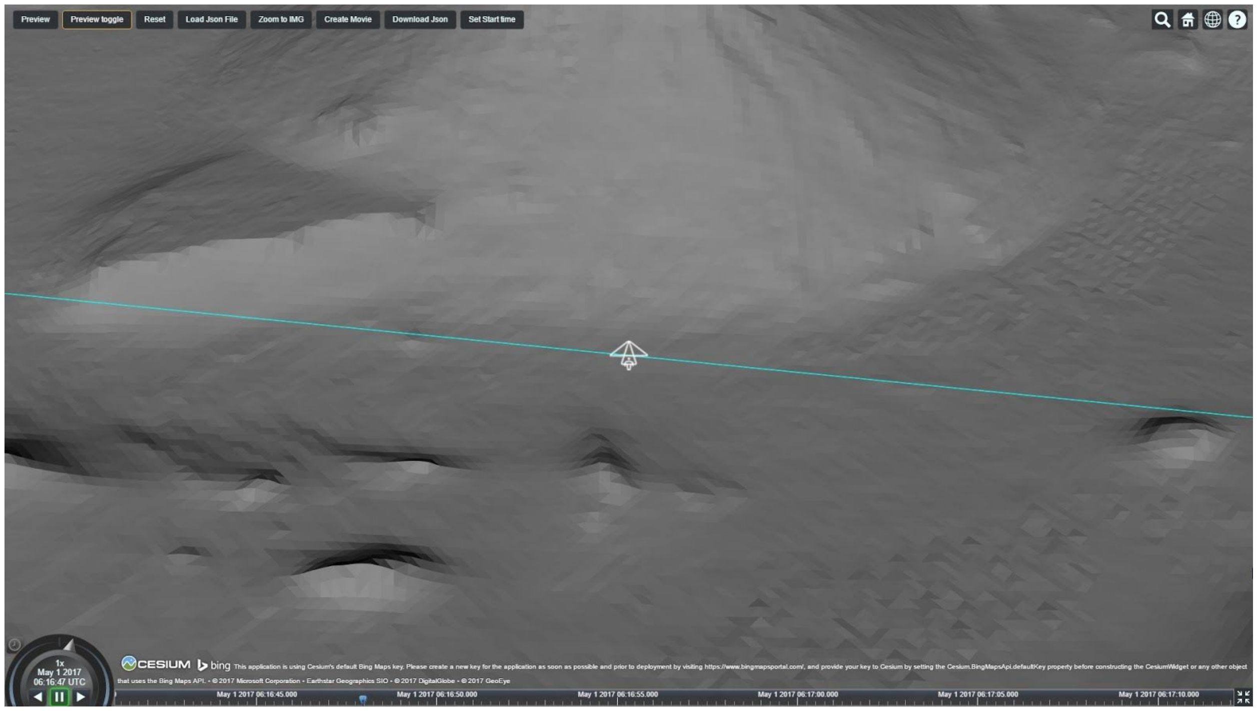
- User Clicks Preview.
- Generates a preview of what the movie is going to look like.



This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • © 2017 Microsoft Corporation • Earthstar Geographics SIO • © 2017 DigitalGlobe • © 2017 GeoEye

Preview Toggle

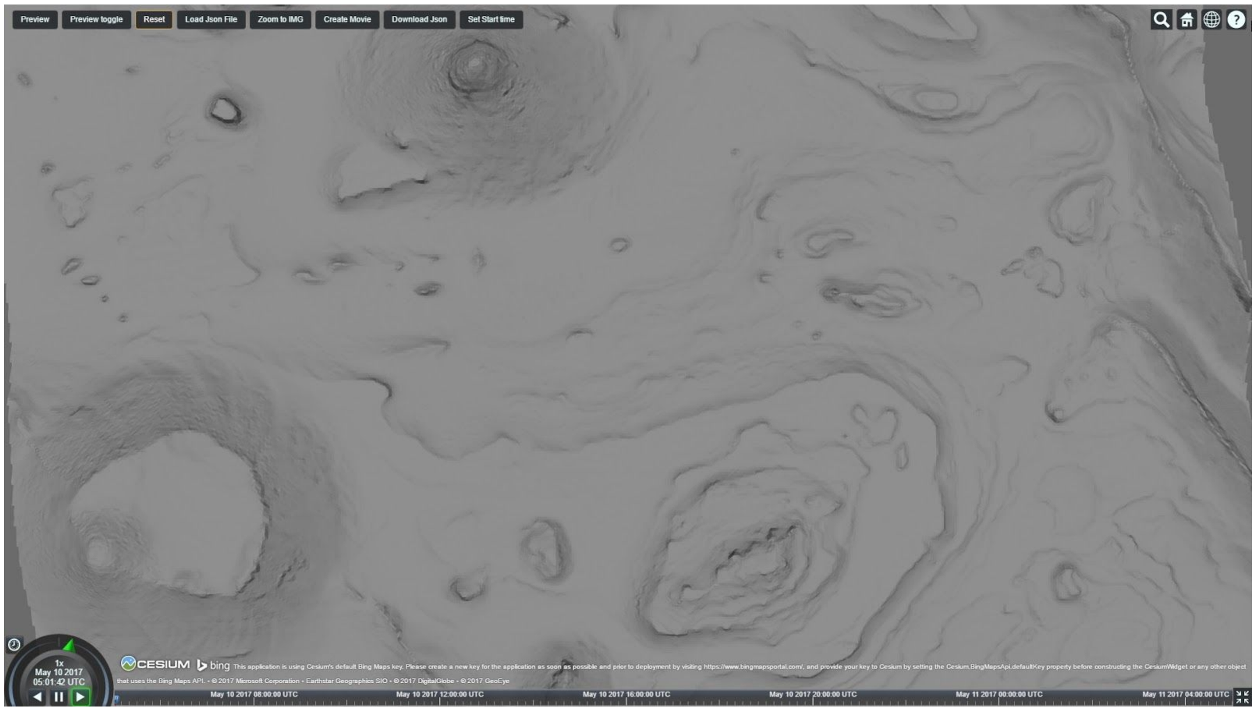
- View the preview in a third person perspective



This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • © 2017 Microsoft Corporation • Earthstar Geographics SIO • © 2017 DigitalGlobe • © 2017 GeoEye

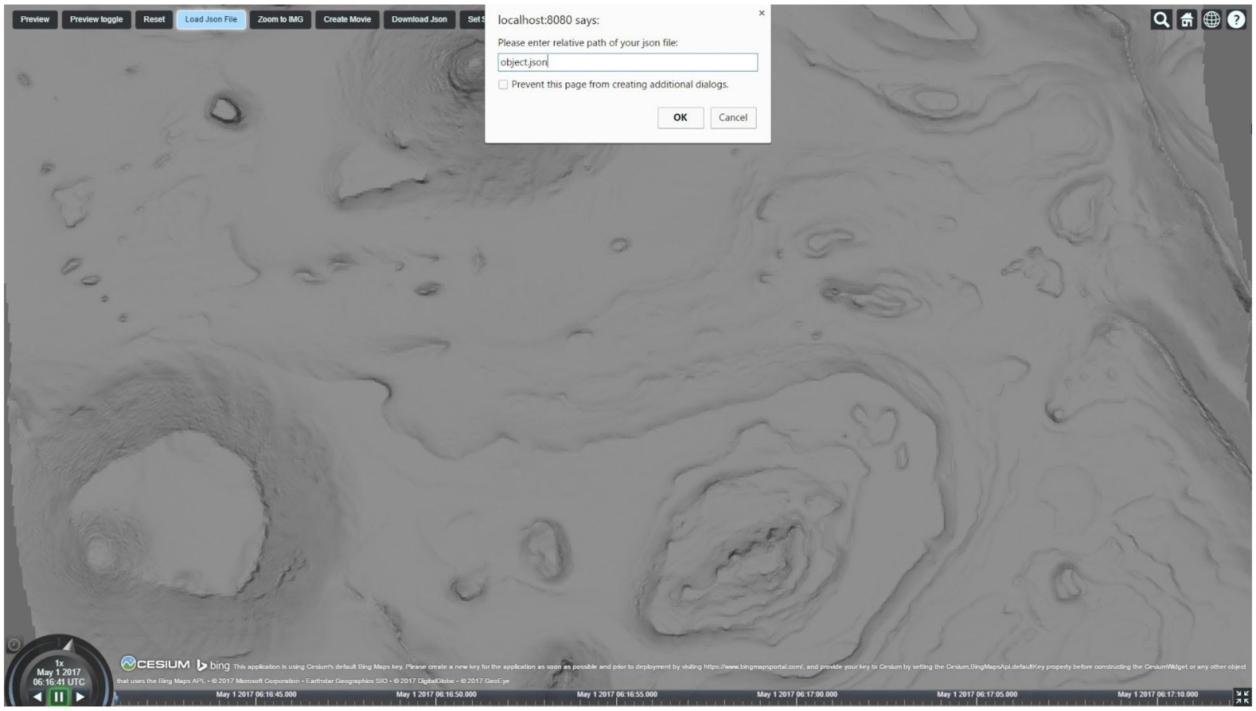
Reset

- Delete all of the points and path



Load Json File

- Load json file from the local machine in relative path
- Enter the path in the textbox (best way is to have the json file in the same directory as the html file)
- Click OK. Points and paths will appear



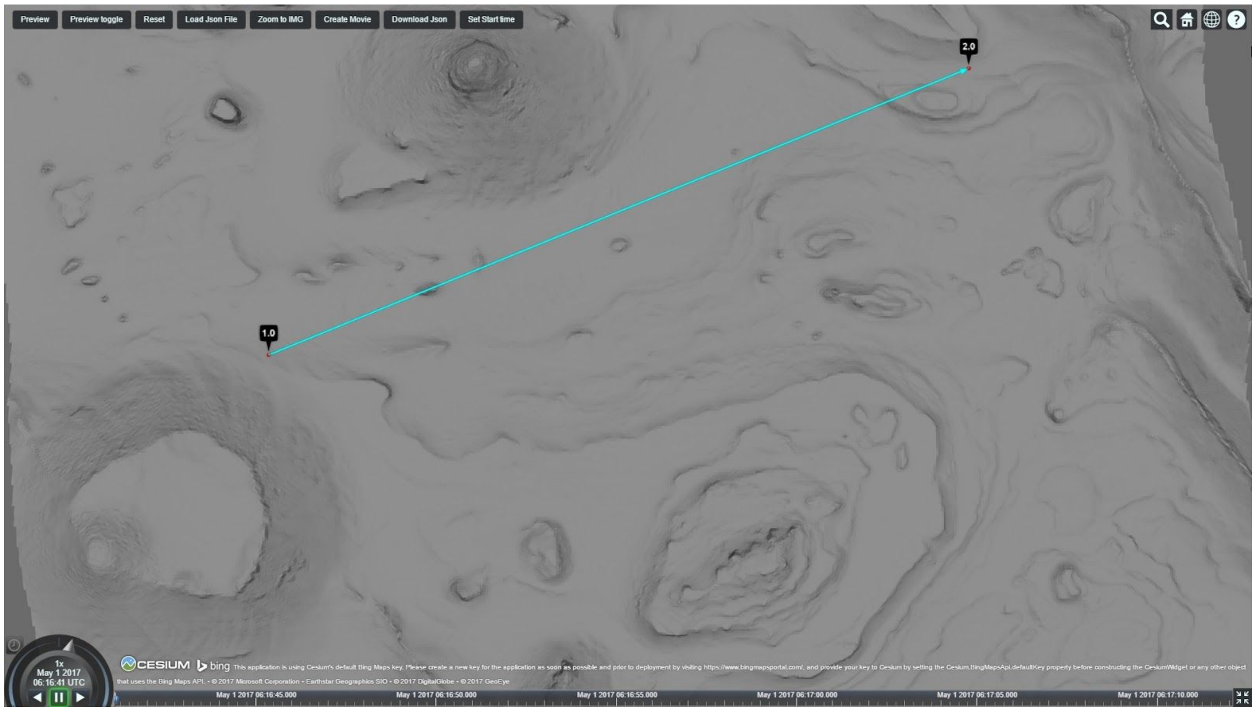
localhost:8080 says:

Please enter relative path of your json file:

object.json

Prevent this page from creating additional dialogs.

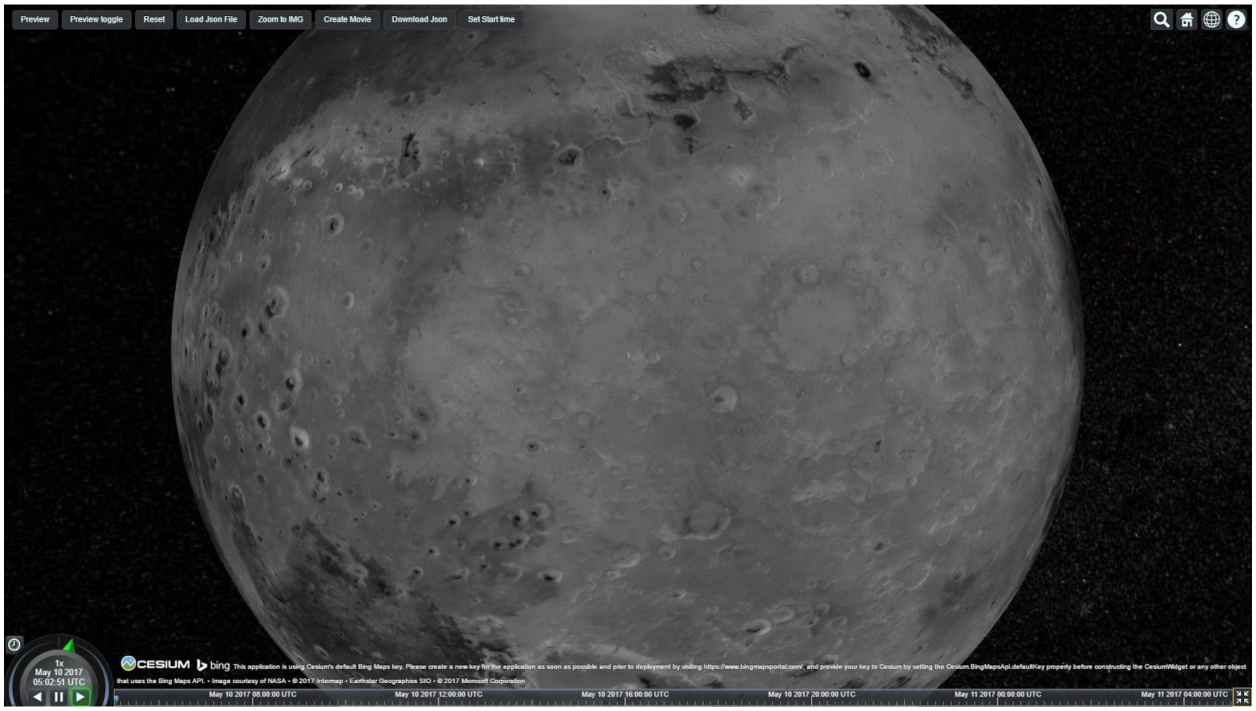
This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • © 2017 Microsoft Corporation • Earthstar Geographics SIO • © 2017 DigitalGlobe • © 2017 GeoEye

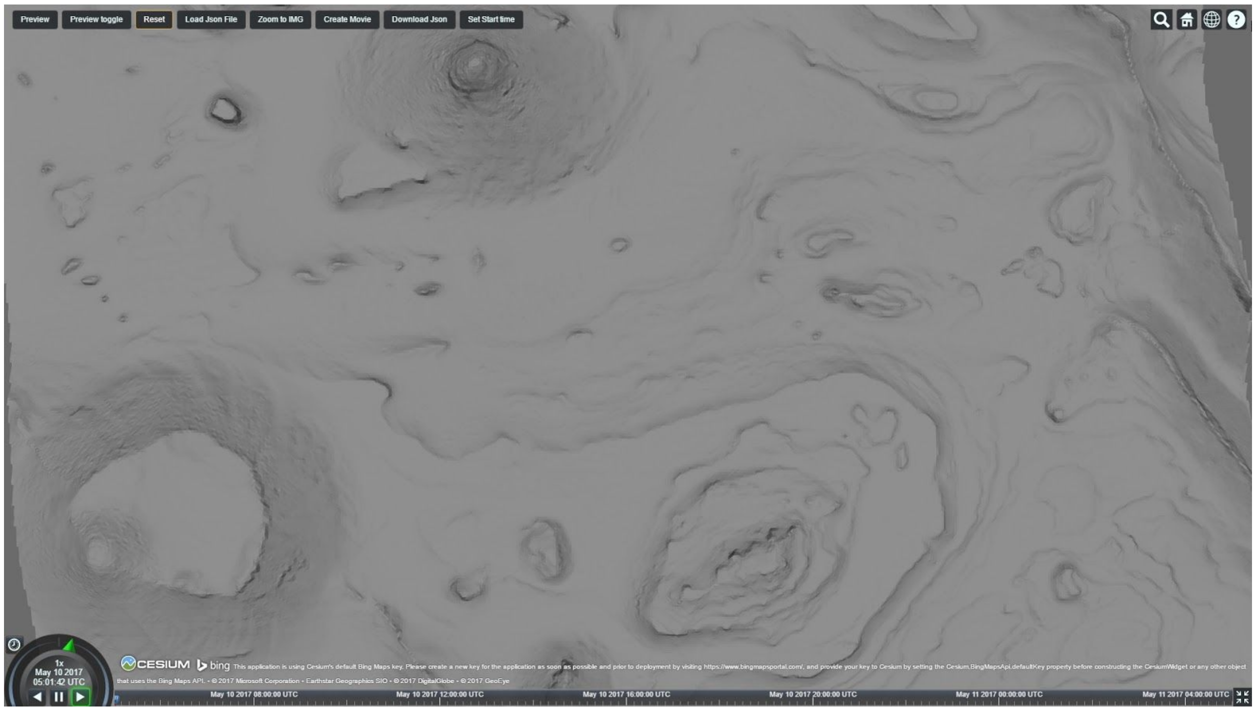


This application is using Cesium's default Bing Maps key. Please create a new key for the application as soon as possible and prior to deployment by visiting https://www.bingmapsportal.com/, and provide your key to Cesium by setting the Cesium.BingMapsApi.defaultKey property before constructing the CesiumWidget or any other object that uses the Bing Maps API. • © 2017 Microsoft Corporation • Earthstar Geographics SIO • © 2017 DigitalGlobe • © 2017 GeoEye

Zoom to IMG

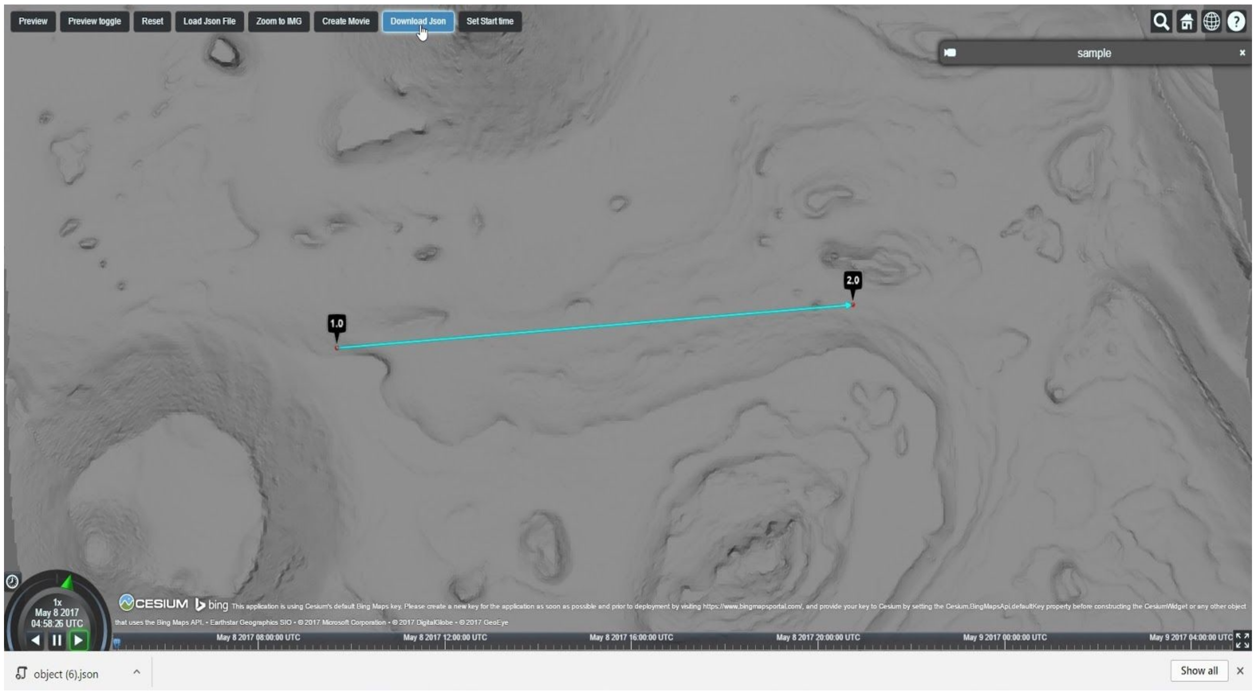
- Zooms into the IMG.





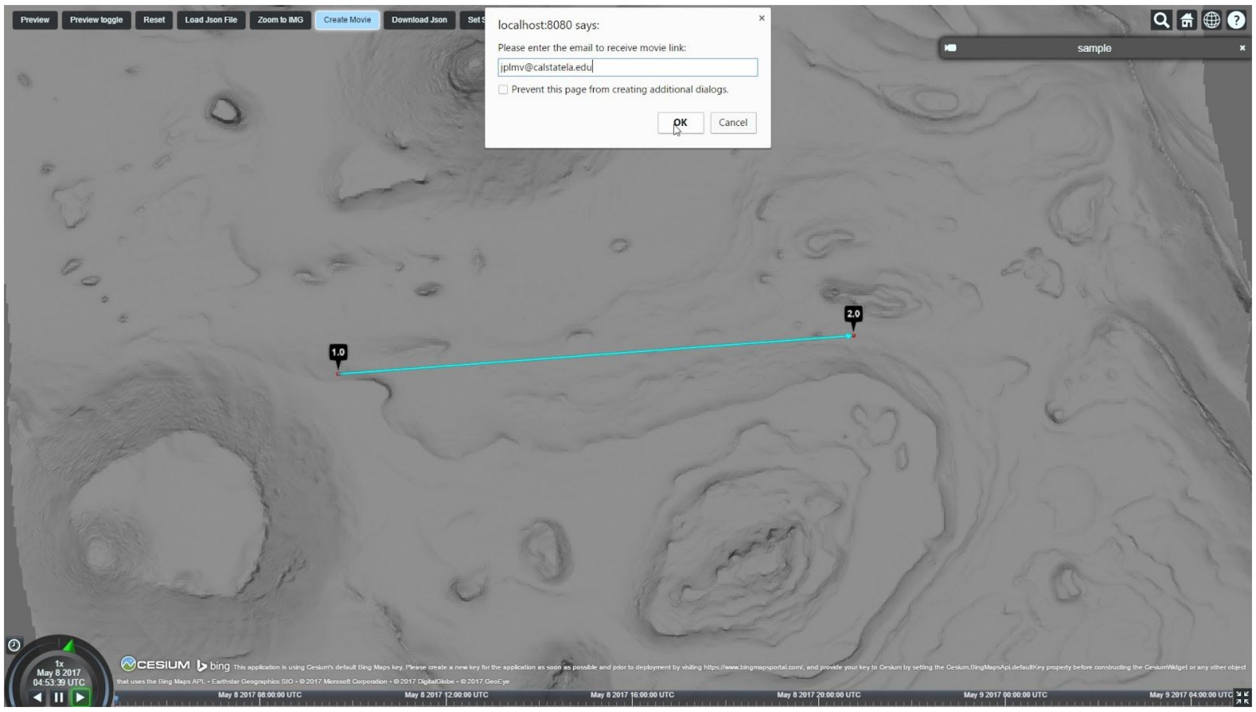
Download Json

- Download all of the user inputs information as a Json file.



Create Movie

- Click create movie
- Enter user email to receive link for final movie.
- Back-end Process begins



localhost:8080 says:

Please enter the email to receive movie link:

jplmv@calstatela.edu

Prevent this page from creating additional dialogs.

4. Installation

Project Download

Download the project directory from CSNS Project page:
http://csns.calstatela.edu/department/cs/project/view?id=5530051

Extract the project.zip file to your home directory.
This document will call this path project_path and you should substitute your full file path in its place.

Inside project_path you should see two directories:
JPLMV
backend

Two files need to have permissions modified to be executable
project/backend/mapper.py
project/backend/reducer.py

Backend Installation

The backend of the system uses the open source application blender for rendering and animating the movie. Our Application and blender depends on several libraries to complete its tasks. The following are the instructions for installing the backend system.

First, Lets install Blender:
Add the ppa to apt so we can grab the current version of blender. You can perform the manual installation by downloading the binaries or source from blender.org. The only requirement is that it is added to the system path so you can execute from command line by typing 'blender'.

Add the PPA to Apt
sudo add-apt-repository ppa:thomas-schiex/blender

Update APT
sudo apt-get update

Install Blender
sudo apt-get install blender

Now that blender is installed, open blender by typing blender from the terminal and set some of the required options.

From the file menu open user preferences (CTRL + ALT + u)
-Editing Tab: New F-Curve Defaults: Change from Bezier --->> Linear
-Add-ons Tab: Enable Import-Export: HiRISE DTM from PDS IMG

If GPU will be used for rendering(requires Nvidia GPU)
-*System Tab: Cycles Compute Device: Set as needed*

Click "Save User Settings" - These settings will now be defaults whenever blender is opened.

A new path will need to be created to hold the blender scripts.

replace username with your username
replace 2.78 with the version of blender you have installed

mkdir /home/username/.config/blender/2.78/scripts/addons/

The scripts for blender need to be copied to this directory

cp project_path/backend/jpl_conf.py
/home/username/.config/blender/2.78/scripts/addons/

repeat this command for animage_scene.py, create_blend.py, create_scene.py, CzmlParser.py

Now Blender is Fully Configured, now the software to support blender needs to be installed.

Install Python 3.5
Python 3.5 is required. This software system has not been tested with Python 2.7 or 3.6.
Install Python 3.5 with development tools, setup tools and package manager
sudo apt-get install python3.5 python3.5-dev python-setuptools python3-pip

Now that the core software is installed it is time to install support libraries for python
First we need to install a software title that allows GDAL bindings in python

Add the UbuntuGIS PPA to Apt
sudo add-apt-repository ppa:ubuntugis/ppa

update APT
sudo apt-get update

install GDAL and python library for GDAL
sudo apt install gdal-bin python3-gdal

Now all that is left is installing python libraries.
sudo pip3 install bottle spiceypy pymongo

Front-end Installation

The backend is completely installed and configured. Now to install the frontend system.

Install NodeJS
curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -
sudo apt-get install -y nodejs

Then open a new terminal, navigate to project_path/JPLMV directory.
Install node dependencies with the following command
npm install

Now the installation is complete. It is now time to start up the server to run the system.

Run the following two commands to start the servers
node server.js (from project/JPLMV directory)
python3 app.py (from project/backend directory)

Now the front end site is accessible through a web browser by using the test link:
localhost:8080/Apps/sandcastle/gallery/front_end_demo.html

5. Conclusion

The Planetary Surface Movie Flyover Generator team succeeded in creating a system architecture to efficiently generate a movie from a user defined camera path and real Mars DEM(Digital Elevation Map) files. Created an easy to use Cesium-based user interface(UI) to create a path and integrate with the existing Mars Trek. Give user a preview of the path generated in first and third person mode. Get Scientifically accurate sun data on each point in time. Created a Server that is able to listen to the UI when ready to initiate the sub-process. Fully Automated System generating a 3D scene that generates a movie file from parsed JSON file. Delivered a rendered video that can be downloaded by the user in the form of a link via the user’s email.

There a few future implementations that we have in mind. We want to integrate more data sets, even though we made the project so that it can use other data sets we have not tested any. We want to implement a more intuitive way to change the altitude. We want to let the user create the path using a polynomial expression. We also want to make further improvements on render times.