Visualization of Orbital Debris with Cesium and Satellite-js

By Daniel A. O'Neil

Introduction

Analytical Graphics Inc. (AGI) provides a free open source digital globe and JavaScript Application Programming Interface (API), named Cesium. With Cesium, a developer can create interactive web-based space mission visualizations. This tutorial explains how to develop a Cesium web-app to visualize orbital debris using the Simple General Perturbations (SGP) model and Two-Line Element (TLE) data.

Prerequisites and Objectives

A TLE encodes a list of orbital elements for Earth-orbiting objects at a given point of time known as an epoch. The SGP is a set of mathematical models for calculating orbital state vectors relative to the Earth-centered inertial coordinate system. The satellite-js code library is an orbital propagator that applies the SGP to TLE to generate coordinates for web-based space mission visualizations. The Center for Space Standards & Innovation operates a web-site, named Celestrak, which provides current TLE data for satellites and orbital debris. The next section of this tutorial explains how to convert a TLE text file to a JavaScript string variable for use by satellite-js SGP orbital propagator functions. Other sections explain how to reference satellite-js and Cesium and present code listings for visualizing and propagating orbital debris.

https://en.wikipedia.org/wiki/Two-line_element_set
https://en.wikipedia.org/wiki/Simplified_perturbations_models
https://github.com/shashwatak/satellite-js
http://www.celestrak.com/NORAD/elements/

Data Conversion

Figure 1 depicts a TLE text file in the background and the same data reformatted as a JavaScript text string. Upon selection of a TLE data-set at Celestrak, the text file will appear in a web-page. Copy and paste the data into Excel. A simple macro can skip the rows and copy the data to another column. The following Visual Basic for Applications (VBA) code snippet copies the data without the label rows.

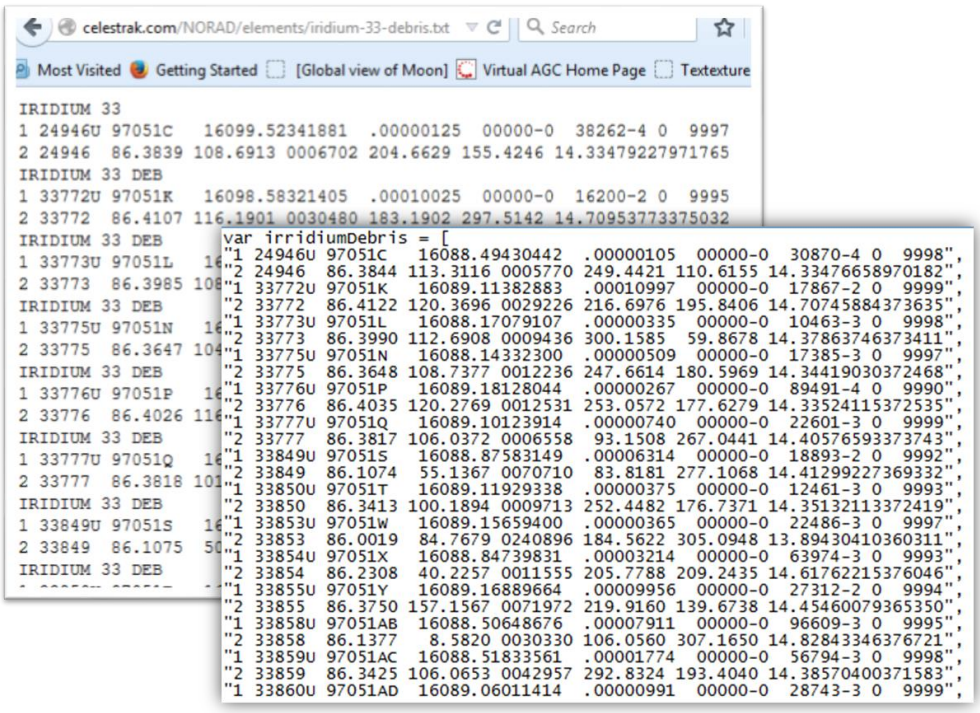

Figure 1 A TLE text file and the same data formatted as a String

```
Sub SkipLabels()
   Dim rgwithLabel As Range
   Dim rgnoLabel As Range
   Dim i As Integer          ' source row counter
   Dim j As Integer          ' destination row counter

   Set rgwithLabel = ThisWorkbook.Worksheets("Labeled").Range("A2")
   Set rgnoLabel = ThisWorkbook.Worksheets("noLabel").Range("A1")
   j = 1
   i = 1
   While rgwithLabel.Cells(i).Value <> ""

     If i Mod 3 <> 0 Then
       rgnoLabel.Cells(j).Value = rgwithLabel.Cells(i).Value
       j = j + 1
     End If
     i = i + 1
   Wend
End Sub
```

Additional reformatting includes adding quotes at the beginning and end of the lines, commas to separate the lines, the variable declaration, and brackets to enclose the strings. The data is saves as a JavaScript file, e.g., Irridium.js. Within the web page for the visualization, the file is included with the line, `<script src="IrridiumDebris.js"></script>`.

Visualization with Cesium

The Satellite-js code library implements the Simple General Perturbations (SGP) model. The Prerequisites and Objectives section provided links to an article about SGP and the Satellite-js GitHub project page. This section provides lines for including the Satellite-js and Cesium code libraries and code snippets for generating the points in Cesium and propagating the orbits with Satellite-js. Trajectories are propagated from TLE data provided by Celestrak. Figure 2 presents a screen-shot of the Cesium and Satellite-js demonstration. The following link leads to the demo.

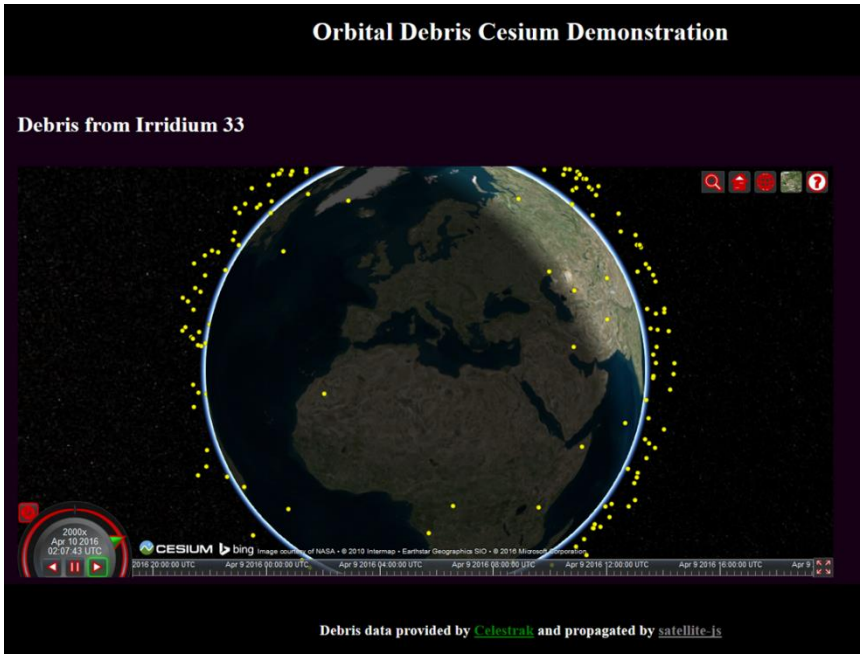

Figure 2 Screenshot of the Cesium and Satellit-js Demo

http://daoneil.github.io/spacemission/Apps/Cesium_with_SGP.html

Referencing the Code Libraries

The previous section provided a line to include a JavaScript file that defines a variable that contains the TLE data. The following lines include the scripts for Cesium and Satellite-js.

```
<script src="../Build/Cesium/Cesium.js"></script>
<link rel="stylesheet" type="text/css" href="satellite-js-
master/sgp4_verification/css/app.css">
<script src="satellite-js-master/dist/satellite.js"></script>
<script src="satellite-js-master/dist/satellite.min.js"></script>
<script src="satellite-js-master/sgp4_verification/lib/angular/angular.js">
</script>
```

JavaScript Code for Cesium

The following code snippet loops through each TLE and adds a point entity to an array. A loop counter becomes the id for the point and an index into the array.

```
  var thing = [] ;
  for (debrisID = 0; debrisID < irridiumDebris.length; debrisID++) {
    thing[debrisID] = viewer.entities.add({
    position : { value : Cesium.Cartesian3.fromDegrees(-75.59777, 40.03883) ,
                 referenceFrame : Cesium.ReferenceFrame.FIXED },
    point : {
            color : Cesium.Color.YELLOW,
            pixelSize : 6
        }});
  }// next debrisID
```

JavaScript Code for Satellite-js

Functions within the Satellite-js library read the TLE and create a record, propagate position based on advancing time, and generate Cartesian coordinates in an Earth Centered Fixed or Earth Centered Inertial reference frame. The following code snippet generates an array of debris records from the TLE data, generates an array of positions and velocities, and propagates the position and velocity array.

```
// Declare orbital debris variables
var debrisRecords = [] ;
var datasetSize = irridiumDebris.length ;
var posVel = [] ;              // positions and velocities of orbital debris

function propagateOrbitalDebris() {
   var j = 0 ;
  for (i=0; i <  datasetSize; i++) {
     var tle1 = irridiumDebris[j] ;
     var tle2 = irridiumDebris[j + 1] ;

    if (typeof tle1 == 'string' || tle1 instanceof String || typeof tle2 ==
        'string' || tle2 instanceof String) {
        debrisRecords[i] = satellite.twoline2satrec(tle1, tle2) ; }
        j = j + 2 ; // advanced to the next TLE in the array }

// Propagate debris using time since epoch
   for (i=0; i <  datasetSize; i++) {
      if (debrisRecords[i] != undefined) { posVel[i] = satellite.sgp4
         (debrisRecords[i], timeSinceTleEpochMinutes); }
   }
```

```
// Propagate debris using time since epoch
for (i=0; i <  datasetSize; i++) {
  if (debrisRecords[i] != undefined) {
     posVel[i] = satellite.propagate(
       debrisRecords[i],
       now.getUTCFullYear(),
       now.getUTCMonth() + 1, // Note, function requires the range 1-12.
       now.getUTCDate(),
       now.getUTCHours(),
       now.getUTCMinutes(),
       now.getUTCSeconds()
    );
  } //endif
}//next i
}//end propateOrbitalDebris
```

Demonstration Source Code

A GitHub repository, https://github.com/daoneil/spacemission, includes the files named Cesium_with_SGP.html and IrridiumDebris.js. The HTML file is the web-page with embedded JavaScript that calls functions in the Cesium and Satellit-js libraries. The JavaScript file defines the variable irridiumDebris, which contains the TLE data for the debris from Irridium 33.

Conclusions

This tutorial provided links to information about SGP, Celestrak, TLE, and Satellit-js. A data conversion section explained to use Excel to format the data as a JavaScript string variable and it include VBA code to copy the data without the headers. A visualization section provided code snippets for referencing the code libraries, creating the point entities that represent the data, and propagating positions based on time advancement. A source code section provides the link to the GitHub repository and identifies the source code files. Potential follow-on development efforts include adding additional debris data sets, changing the point entities to 3D objects, and creating a maneuverable spacecraft for capturing the debris objects.