Senior Design Final Report

Satellite Attitude and Orbit Visualization (SAOV)

Version 1.0 - 04/29/2018

Team Members:

Rolf Castro

Miguel Cayetano

Alan Daniel

Victor Orozco

Allen Ma

Faculty Advisors:

Dr. Sun and Dr. Ye

Liaison:

Sen Yao

Table of Contents

1. Introduction:

1.1. Background

Boeing is a multinational corporation that is known for manufacturing and selling airplanes. They also manufacture and sell other things such as missiles and satellites. Boeing must have a way to accurately track their satellites and wish to explore alternative ways to visualize their active missions. Through Two-Line Element files (TLEs), anyone can translate these files in readable data and read the history of a satellite's position during a specified time. By teaming up with California State University, Los Angeles, Boeing has provided a list of requirements on what they wish to have done in order to experiment with their concept of satellite visualization. Bringing this concept to life could potentially open a pathway to effectively understand a satellite's past, present, and future trajectory. This could also be used as an effective tool to educate others on the subject of satellite movement. If time permitted, Boeing would have had the team use their company's API in order to move away from strictly simulating a scenario in order to dynamically showcase a satellite's current orbit.

What is TLE? In short, TLE is a file that contains two lines of data. This data contains information about an object's orbit around the Earth and serves as a timestamp for this particular object's orbit. There is only one TLE for one object's orbit and this information can be extracted and translated into a three-dimensional visualization to simulate an orbit.

The Satellite Orbit and Attitude Visualization web application is a website that is compatible with multiple browsers such as Firefox, Chrome, and Safari. It takes up to three TLEs and displays the altitude, orbit, and trajectory of each object.

1.2. Design Principles

The software system is a web application that illustrates the position of any satellite at a given time. Controls within the Cesium.js API allow the client to zoom in and out of the display, speedup the orbit, add multiple satellites, and overwrite older satellite two-line element files. The design approach behind the software system includes web programming elements such as websockets in order to communicate to and from the server the site is hosted on. The system is written in JavaScript.

1.3. Design Benefits

The team used JavaScript as the programming language to develop the liaison's desired web application. Alongside JavaScript, the team also implemented Node.js, Cesium.js, Vue.js, and socket.io for generating unique client instances which can update one another's TLE list. JavaScript was the language of choice since the team has knowledge of JavaScript from taking the same class topic. Java was considered but was immediately denied due to the use of Cesium.js.

The Iterative Development approach was used in response to frequent weekly/biweekly meetings. The Waterfall methodology, which the team was familiar with, was replaced with the Scrum methodology. As such, roles were split and each member was responsible for bits and pieces of the application and documents and different demos were generated at times. With different versions of the same application, merge conflicts arose. The team then adapted by performing tests on smaller parts of the application to narrow down bugs. The team also considered a Test-Driven Development approach but the pressure of weekly progress made it difficult to extensively test each part of the application.

1.4. Achievements

Over the course of the academic year, the team has been able to develop an application capable of simulating three satellite orbits at the same time. The original intent of the team's project was to see if Cesium.js was capable of even handling orbital data. As the team progressed further into development, the team discovered that Cesium.js was highly capable of all the things the liaison had been requesting on a weekly basis. Alongside orbital visualization, there are a variety of camera functions that have been implemented to make for a much more dynamic viewing experience. One function allows the users to double-click on the satellite to zoom in and follow it as it travels. Two other camera views allow the users to view Earth as stationary as the orbits happen, the other allows the users to view the orbit as the planet slow spins normally.

This application is also capable of reading TLE data, which can then be translated into a path to display on the Cesium.js-powered website. Over the course of the year, the team had ran into

several problems with browser compatibility and performance issues on machines. The accomplishments from the initial build and the final build allowed for the team to achieve a web application that is browser-friendly and now loads a lot better when given a TLE.

2. Related Technologies

2.1. Existing Solutions

As the team were developing the web application, the team encountered similar sites that serves as a satellite viewer for the public to see; the AGSatTrack and Stuff in Space. Although these applications were already available it was, however, insufficient for Boeing's purposes and desires in terms of the requirements they have provided the team. The team used these existing sites as a reference and a guide as to how the team's application would fare against these robust and well-designed satellite trackers in terms of satellite orbit accuracy, graphical user interface, and ease of use. The team's web application was built upon the use of Cesium.js' geospatial 3D mapping, satellite-js' simplified use of the SGP4/SDP4 calculations, NodeJS as the tema's network model, and Vue.js on top of Cesium.js as the team's front-end to fulfill some of Boeing's requirements.

The reason why Boeing did not utilize AGSatTrack or Stuff in Space was that they were not specific enough to their desires in terms of simulating and visualizing their satellites' orbits and attitudes. AGSatTrack provided satellite stickers and Stuff in Space provided satellites as blips.

These two also did not provide a closer look at the satellite in view, which was required as the team's liaison desired to view the satellites' attitude of individual components. These two applications also provided too much information such as apogee, perigee, inclination, and much more. Whereas the team's liaison requested only the altitude, measured in meters, to be seen in the user interface. These two sites also did not provide their own dropbox for the input of Boeing's proprietary TLEs. This is so that Boeing may visualize and simulate their own satellites within the application the team are developing. The two websites instead provided a cluttered view of satellites orbiting around the earth. Boeing simply wanted to view at most three of their satellites.

The utilization of Cesium.js was only made clear when the team have found out that Cesium's Systems Tool Kit (STK) required contacting someone for permission to use the modeling environment. This was not ideal as, in the beginning and even to this day, the team is no expert in this field whatsoever. The only way to learn more about the development of the project was to do some hands on testing and active research. Cesium.js was perfect for that since it was free, open source, and it contained documentation that could be used when the team hit a roadblock on the development progression. Only through weeks of fiddling the team was able to finally report back to the team's liaison that Cesium.js was sufficient for the project at hand.

2.2. Reused Products

Cesium.js provided the team with a 3D mapping of the Earth to work on. It also provided the team with a time dial to simulate fast-forwarding and reversing of time. This was useful as an

orbital predictor when used in conjunction with the TLEs the team fed into the application. It also provided the team with multiple ways to view the Earth, one of which in particular allows the users to view the satellite up close; the Earth Centered Inertial (ECI). This particular feature is important to properly visualize the attitude of satellite parts when the 3D satellite models' matrix data have been adjusted. The team believed that this can further be used to implement Boeing's proprietary quaternion data which can rotate specific parts of the satellite as needed.

In terms of satellite-js, the team used it to extract necessary orbital data (Like Mean Motion and Epoch) from the TLEs and converted them into a Cesium Language (CZML) file; which is a subset of JSON format for the satellite data.

3. System architecture

3.1. Overview

The application's system architecture is modeled after the MVC (Model-View-Controller) archetype. The Model would represent the back-end which converts the TLEs to CZML files and stores, modifies, and generates satellite models and their orbits. The View would represent Cesium.js and Vue.js, which the team added onto the already existing user interface. The Controller would then be the TLE dropbox and various other options the team have provided as the mediator between the satellite data, satellite model, and the users; basically the mediator between the View and Model.

- **Model (TLE2CZML Module)**

This particular part of the team's application takes the TLE files sent by the user, via the Controller, and converts them to CZML files. It stores the TLEs and CZMLs in a directory back in the server. The Controller reads the newly generated CZML file names and sends it to the View as a list for users to interact with.

- **View (View Module)**

The Cesium.js that the team used was the main component of this section as it provides the visual geospatial 3D mapping of the Earth. With this library the team was able to display satellite orbits and its attitude obtained from the Model via the Controller. The varying options to view the Earth was also provided by Cesium.js and the ability to fast-forward and reverse time was a default setting as well. The user interface is heavily reliant on Cesium.js.

- **Controller (Drag and Drop Module and Server Controller)**

The Controller consists of 2 parts, the Drag and Drop Module and the Server Controller Module. The Drag and Drop Module is a part of the user interface alongside Cesium.js. It simply provides a dropbox for the users to send a TLE of their choosing into the team's Model for conversion. The Server Controller is what communicates between the Model, the View, and any TLEs being sent by the Drag and Drop Module. It also sends the CZMLs chosen to be viewed by the user to Cesium.js in order to view the orbit and attitude of the satellite.

3.2. Data Flow

The following provided describes how the team's web application works. From sending of a TLE file for conversion and back to the users' interactions with the View and the satellite orbits.

- **Drag and Drop Module**

This module is responsible for receiving the users' TLE files. It is necessary as Boeing requires a function in which they are capable of uploading their own proprietary TLEs. By doing so, the users will be capable of visualizing the satellites' orbits and their attitudes with the help of the team's back-end converter, the TLE2CZML Module.

- **Server Controller**

This is responsible for orchestrating between the users' inputs and the server's data. It controls the data flow of the whole application. It sends the TLEs to the TLE2CZML Module and it sends the CZML files to the View Module for the users to view and interact with.

- **TLE2CZML Module**

This module utilizes the satellite-js' Mathematical model SGP4/SDP4 in order to extract information from the densely compressed TLE file. As the name suggests, TLEs are simply two-line-elements which contain data that can be used to accurately model a satellite it is made for. With the use of satellite-js, the TLE can easily be turned into multi-million lines worth of CZML data. The data contained within the CZML is vital to properly display orbital plot points for Cesium to generate. The team also created a CZML converter alongside the use of satellite-js in order to automate the TLE decompression. Once the decompression is complete, both the TLE

and CZML files are then stored within a directory inside the server waiting to be fetched by the users.

- **View Module**

This module takes care of displaying the satellite and the geospatial 3D mapping of Cesium's Earth model. This contains the user interface that the users can interact with and it communicates with the Server Controller to dictate which satellites to be pulled from the server for viewing, what type of view the user would like to see, fast-forward or reverse time for orbital predictions, and works closely with the Drag and Drop Module for TLE uploading.

3.3. Implementation

The project was divided into 2 phases, Phase 1 and Phase 2.

- **Phase 1**

Phase 1 focused on creating a simulation of satellite orbits using TLEs and Cesium. The team found the satellite-js library to be utilized in the server-side for decompression of TLEs and Cesium.js along with Vue.js as the team's front-end for the TLE uploading feature. The display of orbital data was difficult as the plot points are worth multi-million lines of data and very time sensitive. The TLEs are only used for at most 3 days upon generation of the file as the accuracy decays any further than that. Phase 1 also must display at most 3 satellites at a time. This is probably due to the large data sets from the CZML files and the hardware restrictions the team's computers can handle. The altitude of the satellites must also be displayed at any given point in their orbit. This is to be measured in meters. Finally, the application required that the team

allowed users to upload their very own TLE files therefore the team implemented a dropbox for that purpose.

- **Phase 2**

Phase 2 is focuses on Boeing's API of real-time quaternion data. It was meant to adjust the satellite models' matrix data to simulate attitude adjustments to specific parts of the satellite. This was to demonstrate the possible adjustments of orbit and the angle at which the satellite points towards. This is especially important as it is necessary for Boeing's satellites to have their solar panels directed toward the direction of the sun. The team have demonstrated successful solar panel adjustments by altering matrix data within the satellite model itself but Boeing was unable to generate a Non-Disclosure Agreement in time for the team to actually do some attitude adjustment testing. Instead, Phase 2 is desired for future implementations.

4. Conclusions

4.1. Results

The Cesium library has allowed the team to effectively demonstrate orbits in space. The team have developed a web application that can read TLE files, which are then ingested into the application and processed through the Cesium backend. Once the web application processes the files, it successfully generates a satellite's orbit and altitude. The web application takes up to three TLE files.

The team made use of various public packages found on github, such as tle2czml and satellite-js in order to successfully extract information from the TLE files that the team's application can then understand.

The View Module handles the display of everything processed through the backend of the application. This is where the CZML component exits after TLE processing.

These parts are then combined in order to provide a visualization for a satellite orbiting around the Earth for a point in time that has already happened.

4.2. Future

Although significant progress was made in terms of simulating satellites orbiting around the Earth, every TLE file has a lifespan of 3 days. The team wishes to pass on the project to future teams so that it may eventually read real-time data. The current application only shows stationary satellites moving about as points in space and it would be of great interested to implement dynamic satellite behavior in order the follow-up with Boeing's interest of a satellite visualization tool. The following are suggestions to next year's group:

- Implementing Boeing's API in order to retrieve real-time satellite data
- Implementing satellite behaviors so that the visualization follows the behavior of the real satellite with little delay.

5. References

Cesium.js API: https://cesiumjs.org/refdoc/

satellite-js repository: https://github.com/shashwatak/satellite-js

MMT Observatory on TLE files: https://www.mmto.org/obscats/tle.html

TLE2CZML repository: https://github.com/kujosHeist/tle2czml