A 3D Satellite Orbit Viewer for the Web
A web-based tool for satellite orbit prediction and visualization built with WebGL

V1.0 – 6 March 2014
Table of Contents

1 Background ............................................................................................................................................. 3
2 Project Vision......................................................................................................................................... 3
3 Project scope ......................................................................................................................................... 4
4 Architectural requirements .................................................................................................................... 4
5 Deliverables.......................................................................................................................................... 5
6 Skills requirements ............................................................................................................................... 6
7 Supplemental information ...................................................................................................................... 6
8 About Pinkmatter .................................................................................................................................. 6
1 Background

There are currently over 2000 man-made satellites orbiting the earth. Their purpose ranges from telecommunication to atmospheric monitoring to earth imaging for environmental management and even spying. The orbits of these satellites can take various shapes, times and altitudes and are subject to the Keplerian laws of planetary motion.

There are a number of tools out there that can visualize satellite orbits (i.e. plot them on a map) based on their Two Line Element (TLE) description: a standard notation that provides all necessary orbital parameters. The purpose of this project, however, is to build a viewer that can do so in 3D inside a web-browser.

![Figure 1: A 2D orbit viewer showing the ground tracks, positions and footprints of two satellites, Landsat 7 and 8](image)

2 Project Vision

Although there are a number of tools out there that can visualize satellite orbits, many of them do so only in two dimensions. Those that support 3D views (e.g. Google Earth – yes, it can display satellite positions as they orbit the earth) typically require an installation on the client’s PC or are not very flexible. With the (relatively recent) advent of the WebGL standard, 3D animation is now a reality for most web browsers.

This project aims to build a 3D satellite orbit visualization tool that can be run with a web browser alone, thereby making it available to a much wider audience. The purpose of a tool like this is to enable education, inspire the general public and also to produce a library that can be embedded into other more complex web applications.
3 Project scope

The 3D Orbit Viewer shall be built as a single-page web application that displays the following:

- The earth as an ellipsoid with a textured surface derived from a low-resolution image
- The day/night line based on the current sun position
- The orbital path of the chosen satellites
- The current position of the selected satellites as a rough geometrical model
- The position of satellite ground stations painted on the earth’s surface
- The ground track of the satellites (orbital path projected onto the earth surface)
- The current area of the earth that is imaged by the satellite, highlighted/painted on the surface. This can be represented by highlighted squares or by a JPEG painted onto the surface of the globe.

Furthermore the application will allow the user to interact with the view as follows:

- Change the camera angle to one of a number of presets (e.g. eastern hemisphere, western hemisphere, north pole, south pole)
- Choose different view modes, e.g.
  - Keep one satellite in center (and rotate the earth)
  - Keep a point on the ground in center (based on a geographic coordinate)
  - Keep the earth in a fixed view (and animate all the satellites)
  - Rotate the earth and the satellites
- Zoom into areas to display, for example, only a quarter of the globe. Zooming does not have to update the underlying texture (no need to rebuild Google Earth).
- Pause/resume the animation
- Set the time (i.e. current time or some other date/time)
- Set the time resolution (i.e. real time, 2x real time, etc.)
- Select a satellite or ground station and get information about it in a side panel

The following aspects of the application should be configurable (with a configuration file on the server):

- The IDs (NORAD codes) of the satellites to be rendered
- Configuration of each satellite (colour, angle of view, etc.)
- The location of TLE files (local directory on the server)
- The image to use as the earth texture
- The view mode in which the application starts up (camera angle, zoom level, animation mode, time, etc.)

4 Architectural requirements

Overview

As a web application, the Orbit Viewer will consist of a backend server with a web frontend. The backend server performs the calculations while the web frontend renders the globe and satellite orbits.
Technologies

Server
- Implemented in Java
- Compatible with standard Java web containers like Tomcat, Jetty and Glassfish

Client
- Single page web application using HTML and JavaScript
- WebGL for 3D rendering
- Compatible with Internet Explorer 11 and Chrome v40+ (on Windows and Linux)

Performance
Performance of the application should be adequate to provide a good user experience on commonly-used desktop and laptop hardware. This implies smooth and jitter-free animations.

Design principles
Good object orientated design principles and best practices are a critical requirement for this application. The application needs to be assembled from well-defined objects that each perform a clearly defined function. An approach using dependency injection is preferred. This should lead to well separated code, commonly known as “ravioli code” – in contrast to the spaghetti and lasagne that is produced by poor design principles.

Clear API
The application should be developed as a library to be embedded into other applications. It should be kept in mind that this single page web application is a just a demonstrator of the underlying 3D Orbit Viewer library being developed.

Libraries
External libraries can be used where applicable. The project, however, should not depend on any GPL libraries. LGPL or other less strict license agreements (Apache, CCDL, BSD, etc.) are acceptable.

5 Deliverables
The following component are required as the deliverables of this project:

- Source code of server and client components
- Installation and user documentation (in English)
- Architecture documentation and diagrams detailing the core components and their interdependencies
- API documentation (Javadoc)
6 Skills requirements

Teams tendering for this project should have:

- A good background in mathematics
- Solid Java skills
- JavaScript skills
- An interest in 3D rendering
- An interest in space sciences
- A passion for coding and software engineering

7 Supplemental information

Introduction to orbital mechanics


What is WebGL

- [https://dev.opera.com/articles/introduction-to-webgl-part-1/](https://dev.opera.com/articles/introduction-to-webgl-part-1/)

8 About Pinkmatter

Pinkmatter Solutions is a software company based in Pretoria, South Africa. The team consists of engineers and scientists and provides specialized software services to space agencies across the globe. Our client base ranges from leading space agencies to smaller receiving stations in emerging countries.

This project will be supervised by:

Chris Böhme
chris@pinkmatter.com

Pinkmatter staff will assist with design decisions and provide technical inputs. Weekly meetings will be held remotely (Skype, etc.). A face to face meeting will be held every two weeks. Petrol allowance is available for travel to Pinkmatter’s offices.