Satellite Orbit Tracker

Orbital calculations and 3D visualization entirely in the browser with JavaScript.

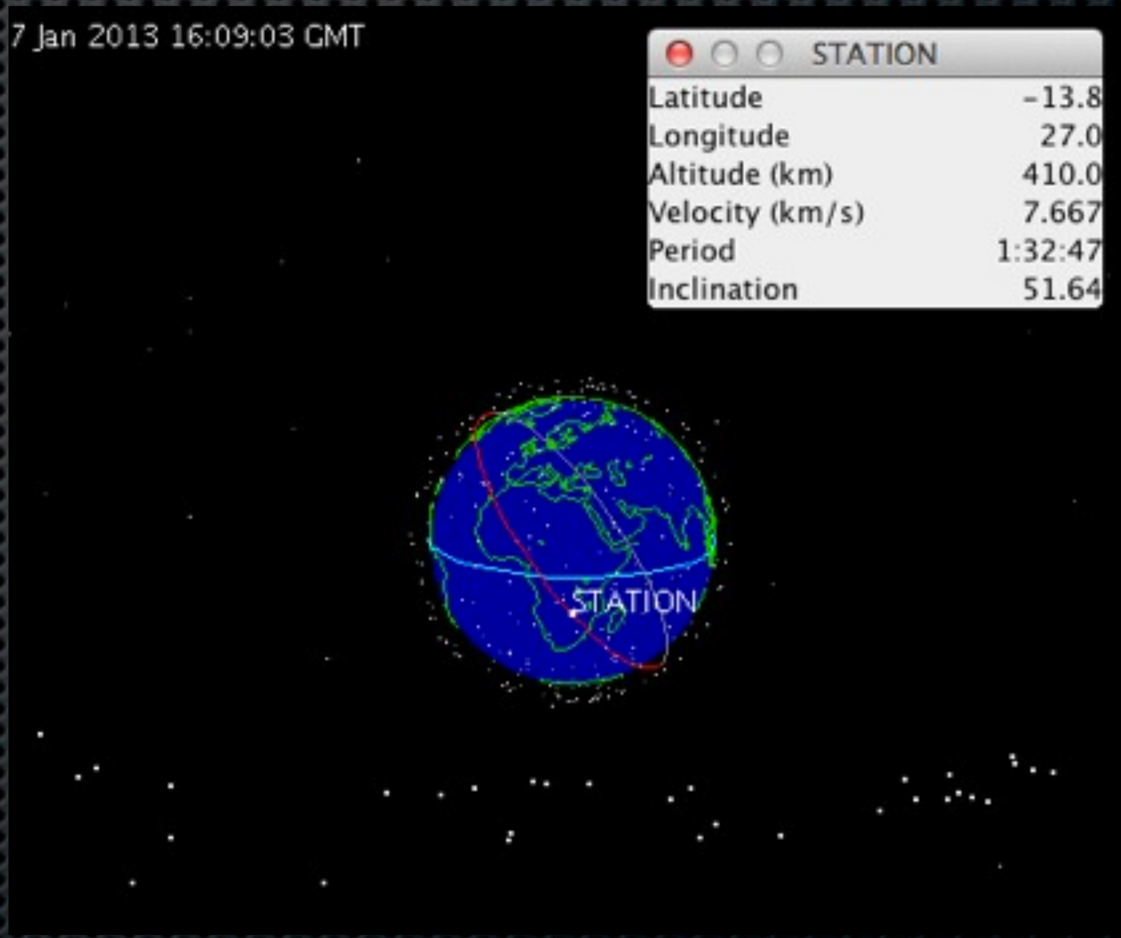
A modern approach to interactive web applications.

2013-01-08 Presentation to HITSS AppDev

Chris Shenton <chris@koansys.com> <chris.shenton@nasa.gov>

In the Beginning: JTrack3D

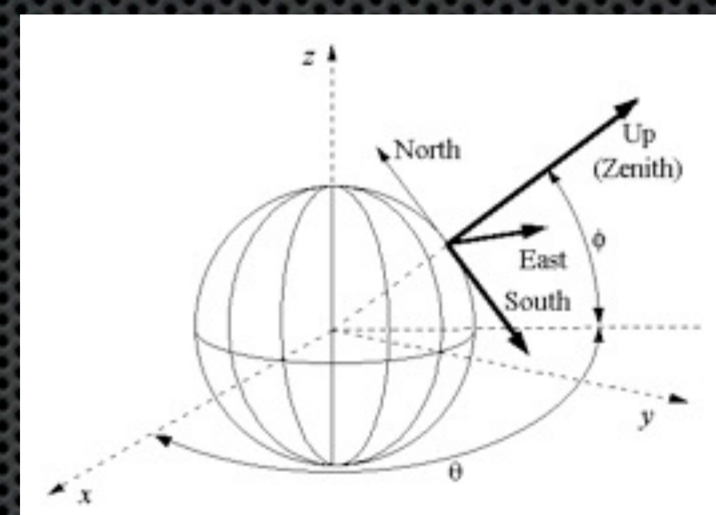
- > 10 years old applet at MSFC
- Migrated from MSFC to HQ
- Resurrected by decompiling
- Java: not working well now
- Educators disappointed

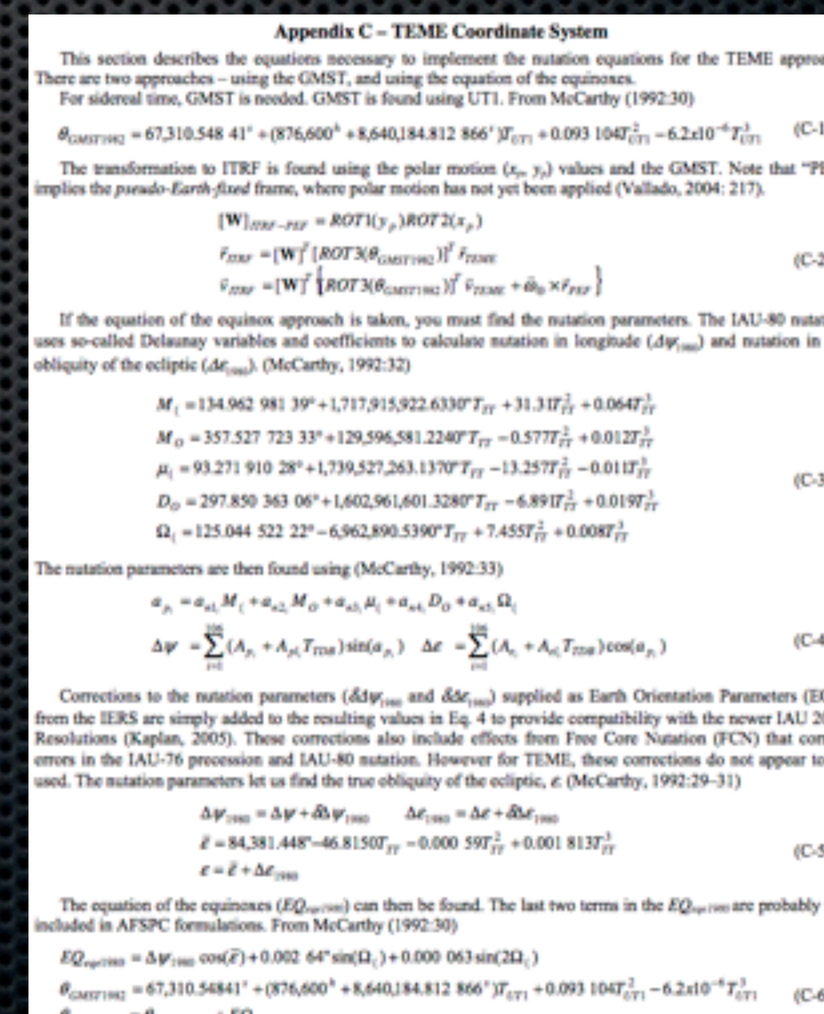


Java no longer installed by Apple. 64-bit Java from Oracle won't run in Chrome as it's 32-bit; have to use FireFox, Safari. When we migrated Science.nasa.gov from MSFC to HQ's NASAscience.nasa.gov we were initially told not to bother with JTrack3D due to the delay it would have caused.

SGP: Orbit Calculations

- Simplified General Perturbation model
- Developed in 1959
- FORTRAN in 1988
- MATLAB, others
- Miura's PhD thesis in 2009
- TLEs used by NORAD, NASA





Appendix C – TEME Coordinate System

This section describes the equations necessary to implement the nutation equations for the TEME approach. There are two approaches – using the GMST, and using the equation of the equinoxes.

For sidereal time, GMST is needed. GMST is found using UT1. From McCarthy (1992:30)

$$\theta_{GMST1982} = 67{,}310.548\ 41^s + (876{,}600^h + 8{,}640{,}184.812\ 866^s)T_{UT1} + 0.093\ 104T_{UT1}^2 - 6.2\times10^{-6}T_{UT1}^3 \quad \text{(C-1)}$$

The transformation to ITRF is found using the polar motion (x_p, y_p) values and the GMST. Note that "PEF" implies the *pseudo-Earth-fixed* frame, where polar motion has not yet been applied (Vallado, 2004: 217).

$$\begin{aligned} [\mathbf{W}]_{ITRF-PEF} &= ROT1(y_p)ROT2(x_p) \\ \vec{r}_{ITRF} &= [\mathbf{W}]^T [ROT3(\theta_{GMST1982})]^T \vec{r}_{TEME} \\ \vec{v}_{ITRF} &= [\mathbf{W}]^T \left\{ [ROT3(\theta_{GMST1982})]^T \vec{v}_{TEME} + \vec{\omega}_\oplus \times \vec{r}_{PEF} \right\} \end{aligned} \quad \text{(C-2)}$$

If the equation of the equinox approach is taken, you must find the nutation parameters. The IAU-80 nutation uses so-called Delaunay variables and coefficients to calculate nutation in longitude ($\Delta\psi_{1980}$) and nutation in the obliquity of the ecliptic ($\Delta\varepsilon_{1980}$). (McCarthy, 1992:32)

$$\begin{aligned} M_{\leftmoon} &= 134.962\ 981\ 39^\circ + 1{,}717{,}915{,}922.6330''T_{TT} + 31.310T_{TT}^2 + 0.064T_{TT}^3 \\ M_{\odot} &= 357.527\ 723\ 33^\circ + 129{,}596{,}581.2240''T_{TT} - 0.577T_{TT}^2 + 0.012T_{TT}^3 \\ \mu_{\leftmoon} &= 93.271\ 910\ 28^\circ + 1{,}739{,}527{,}263.1370''T_{TT} - 13.257T_{TT}^2 - 0.011T_{TT}^3 \\ D_{\odot} &= 297.850\ 363\ 06^\circ + 1{,}602{,}961{,}601.3280''T_{TT} - 6.891T_{TT}^2 + 0.019T_{TT}^3 \\ \Omega_{\leftmoon} &= 125.044\ 522\ 22^\circ - 6{,}962{,}890.5390''T_{TT} + 7.455T_{TT}^2 + 0.008T_{TT}^3 \end{aligned} \quad \text{(C-3)}$$

The nutation parameters are then found using (McCarthy, 1992:33)

$$\begin{aligned} a_{p_i} &= a_{n1_i}M_{\leftmoon} + a_{n2_i}M_{\odot} + a_{n3_i}\mu_{\leftmoon} + a_{n4_i}D_{\odot} + a_{n5_i}\Omega_{\leftmoon} \\ \Delta\psi &= \sum_{i=1}^{106}(A_{p_i} + A_{p1_i}T_{TDB})\sin(a_{p_i}) \quad \Delta\varepsilon = \sum_{i=1}^{106}(A_{e_i} + A_{e1_i}T_{TDB})\cos(a_{p_i}) \end{aligned} \quad \text{(C-4)}$$

Corrections to the nutation parameters ($\delta\Delta\psi_{1980}$ and $\delta\Delta\varepsilon_{1980}$) supplied as Earth Orientation Parameters (EOP) from the IERS are simply added to the resulting values in Eq. 4 to provide compatibility with the newer IAU 2000 Resolutions (Kaplan, 2005). These corrections also include effects from Free Core Nutation (FCN) that correct errors in the IAU-76 precession and IAU-80 nutation. However for TEME, these corrections do not appear to be used. The nutation parameters let us find the true obliquity of the ecliptic, ε (McCarthy, 1992:29–31)

$$\begin{aligned} &\Delta\psi_{1980} = \Delta\psi + \delta\Delta\psi_{1980} \qquad \Delta\varepsilon_{1980} = \Delta\varepsilon + \delta\Delta\varepsilon_{1980} \\ &\bar{\varepsilon} = 84{,}381.448'' - 46.8150T_{TT} - 0.000\ 59T_{TT}^2 + 0.001\ 813T_{TT}^3 \\ &\varepsilon = \bar{\varepsilon} + \Delta\varepsilon_{1980} \end{aligned} \quad \text{(C-5)}$$

The equation of the equinoxes ($EQ_{equinox1980}$) can then be found. The last two terms in the $EQ_{equinox1980}$ are probably not included in AFSPC formulations. From McCarthy (1992:30)

$$\begin{aligned} EQ_{equinox1980} &= \Delta\psi_{1980}\cos(\bar{\varepsilon}) + 0.002\ 64''\sin(\Omega_{\leftmoon}) + 0.000\ 063\sin(2\Omega_{\leftmoon}) \\ \theta_{GMST1982} &= 67{,}310.54841^s + (876{,}600^h + 8{,}640{,}184.812\ 866^s)T_{UT1} + 0.093\ 104T_{UT1}^2 - 6.2\times10^{-6}T_{UT1}^3 \\ \theta_{GAST1982} &= \theta_{GMST1982} + EQ_{equinox1980} \end{aligned} \quad \text{(C-6)}$$

```
ISS (ZARYA)
1 25544U 98067A   12312.59429472  .00006293  00000-0  11296-3 0  6057
2 25544  51.6473 129.0708 0015313 255.0735 149.5553 15.51208285800293
```


SGP: from a known point, velocity, etc, calculate forward in time to next position, velocity, etc; repeat. Math from: http://celestrak.com/columns/v02n02/, http://celestrak.com/publications/AIAA/2006-6753/. The SATRAK code used by JSC is SECRET due to NORAD restrictions.

Miura's thesis indicated the MATLAB implementation was the most accurate, so this is what I started with. TLE: Two line element set giving last verified position, velocity, time, other math attributes.

JavaScript is Powerful, Fast

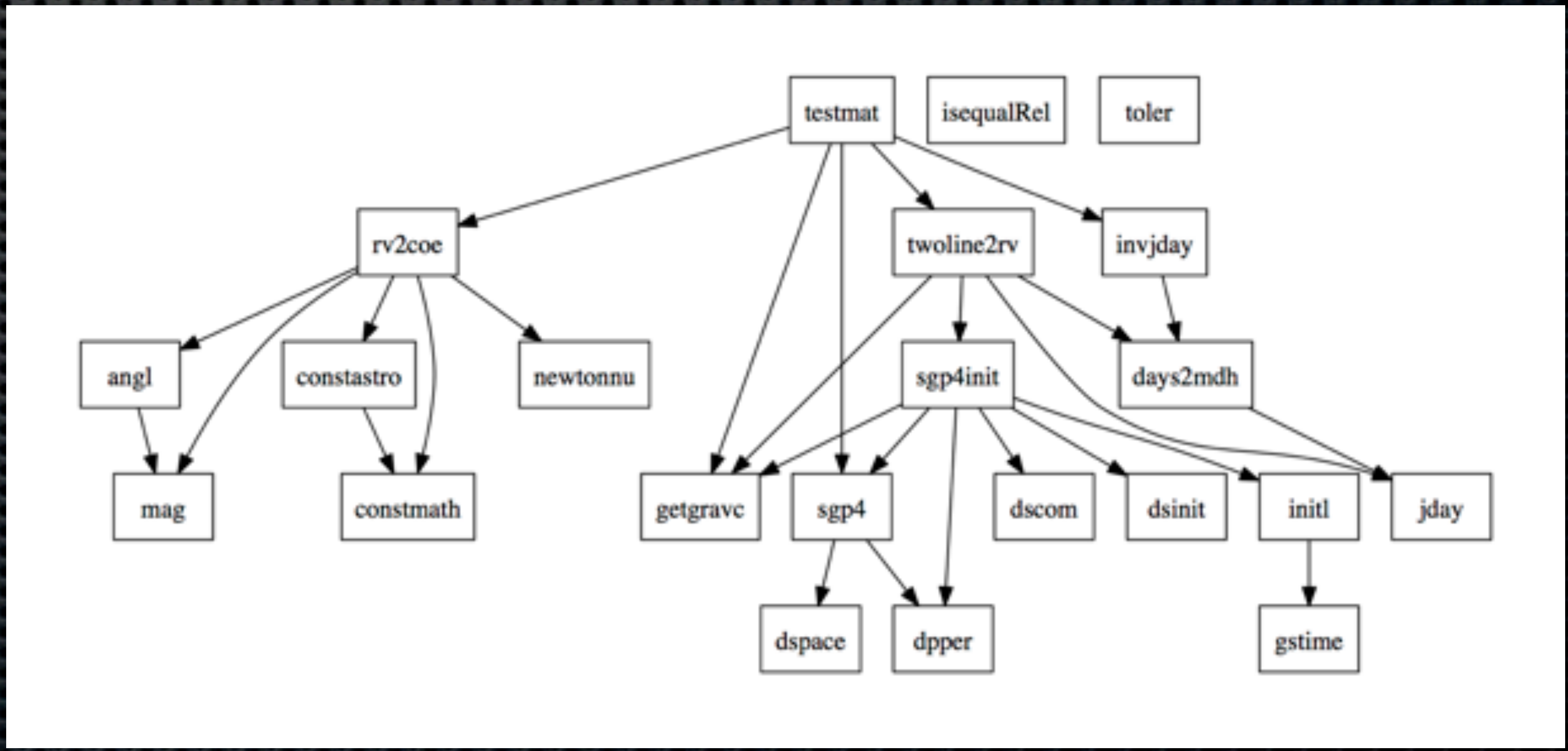
- JavaScript is the language of the web
- Run entirely in browser, no servers needed
- Performance isn't everything, but matters here
- JavaScript is expressive, a real (powerful) language

Language	CPU time			Slower than	
	User	System	Total	C++	previous
C++ (*optimized with -O2*)	1,520	0,188	1,708	-	-
Java (*non-std lib*)	2,446	0,150	2,596	52%	52%
C++ (*not optimized*)	3,208	0,184	3,392	99%	31%
Javascript (nodejs)	4,068	0,544	4,612	170%	36%
Java	8,521	0,192	8,713	410%	150%
Python	27,886	0,168	28,054	1543%	108%
Perl	41,671	0,100	41,771	2346%	49%
PHP 5.3	94,622	0,364	94,986	5461%	127%

Comparison table abstracted from
http://blog.famzah.net/2010/07/01/cpp-vs-python-vs-perl-vs-php-performance-benchmark/

Translating MATLAB to JS

- Problem: I didn't know MATLAB
- Problem: I didn't know JavaScript
- Graph dependencies
- Translate literally

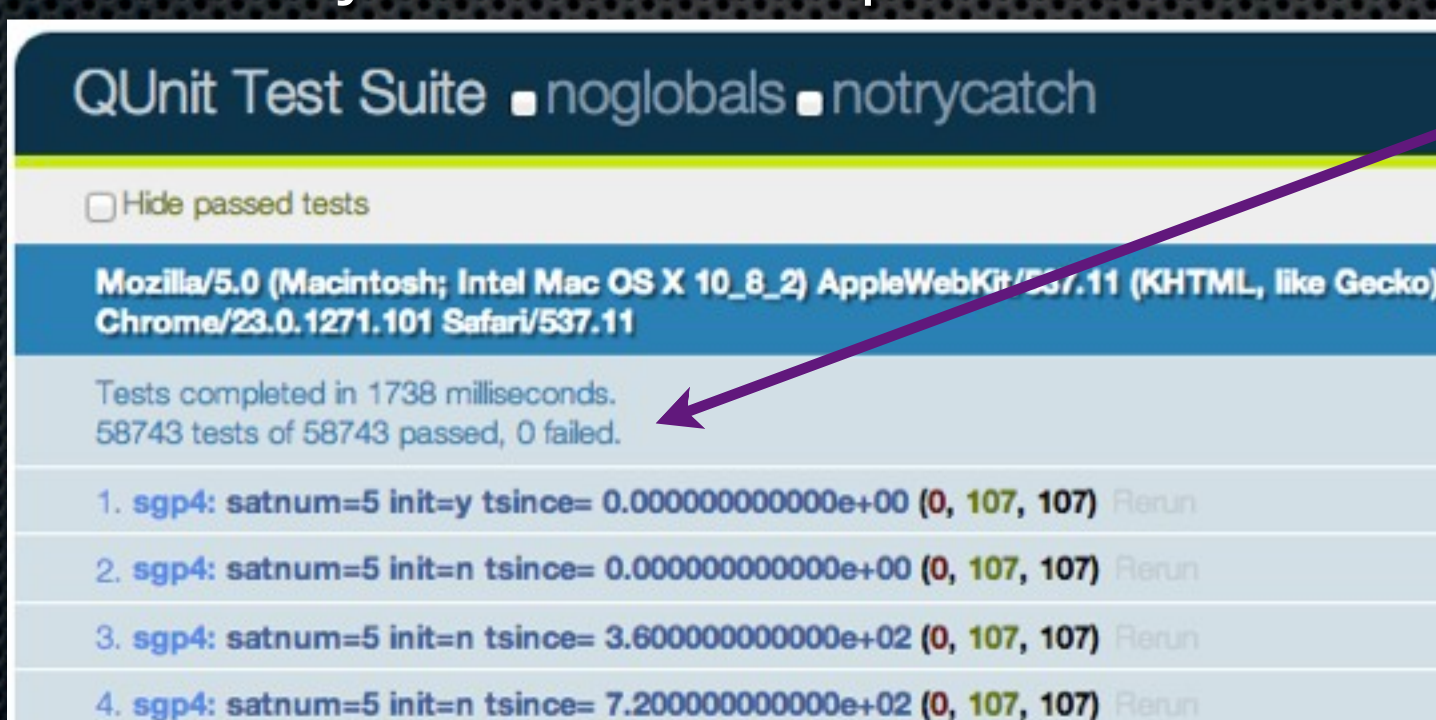


Tuesday, January 8, 13

Write a source code dependency grapher to show what MATLAB modules depended on which others, then translate each function, one by one. Use literal -- rather than idiomatic -- translation to avoid introducing errors.

Tests: Only Way to Verify

- Create tests for MATLAB, capture results
- Recreate for JavaScript to verify results
- Single module had almost 60,000 tests
- Exercised all other modules
- I told you JavaScript was fast: 1.738 seconds



Tuesday, January 8, 13

That's 38,000 tests per second exercising all the math modules!

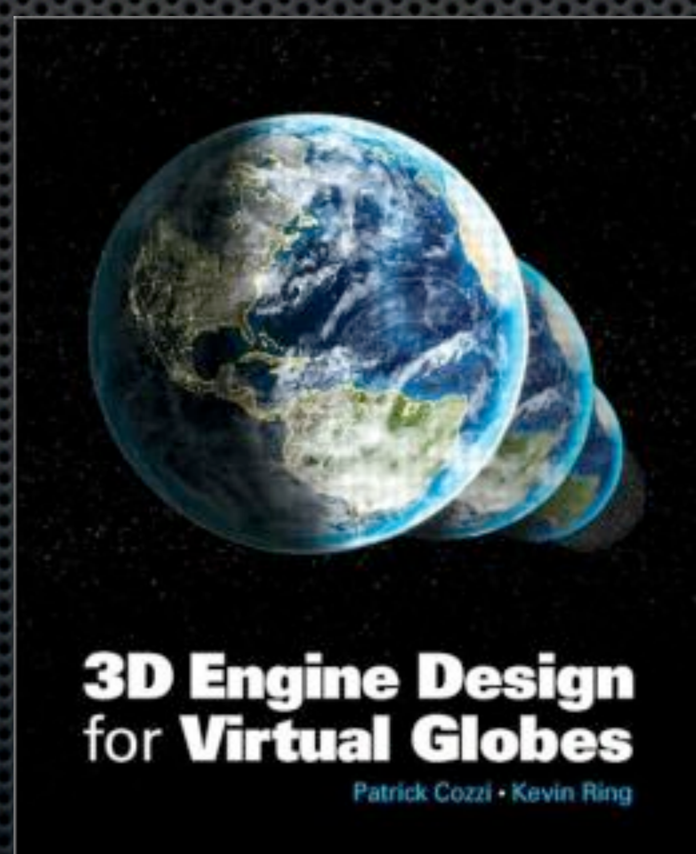
Visualization?

- How do we visualize the computed positions? Earth?
- Lots of libraries in various states of readiness
- I don’t want to have to invent 3D visualization :-(
- I don’t know how to invent 3D visualization!

JS game, graphics, physics engines table: http://www.reddit.com/r/javascript/comments/f094j/list_of_js_game_engines_community_effort

Cesium

- A chance meeting at NASA Open Source Summit 2012
- AGI has lots of smart PhDs
- Open Source
- Pure JavaScript
- WebGL-based



Tuesday, January 8, 13

Check out AGI's mind-blowing "Lots of Satellites" live demo; the position data is calculated by servers in the cloud, streamed to the browser-based Cesium rendering engine. Code: https://github.com/AnalyticalGraphicsInc/cesium

WebGL-Capable Platforms

- Desktops supported now on most browsers
- Mobile coming along quickly

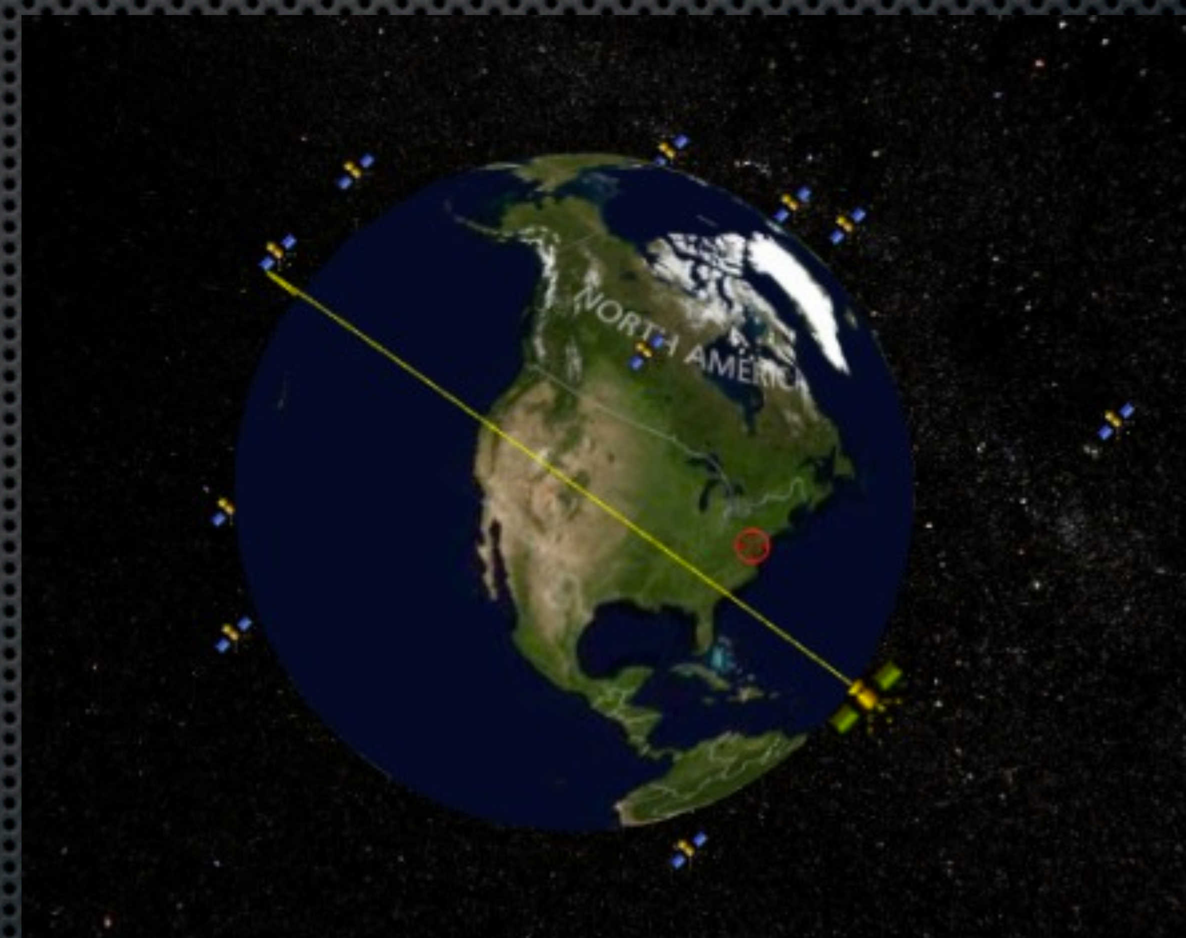
Desktop	Android	iOS
IE: with ChromeFrame FireFox: yes Chrome:yes Safari: if enabled	Default Browser: no FireFox: yes Chrome:?	No: disabled by Apple except for advertising partners :-(

Tuesday, January 8, 13

Recent Android (e.g., Nexus 7) with recent FireFox. I'd be surprised if Apple didn't enable WebGL soon, probably testing with advertising partners first.

SOT: Satellite Orbit Tracker

- Calculate with SGP
- Visualize with Cesium
- All JavaScript
- Entirely in Browser
- Danger: live demo 



Code: https://github.com/koansys/sot
Running Previews: http://sot.koansys.com/viz/