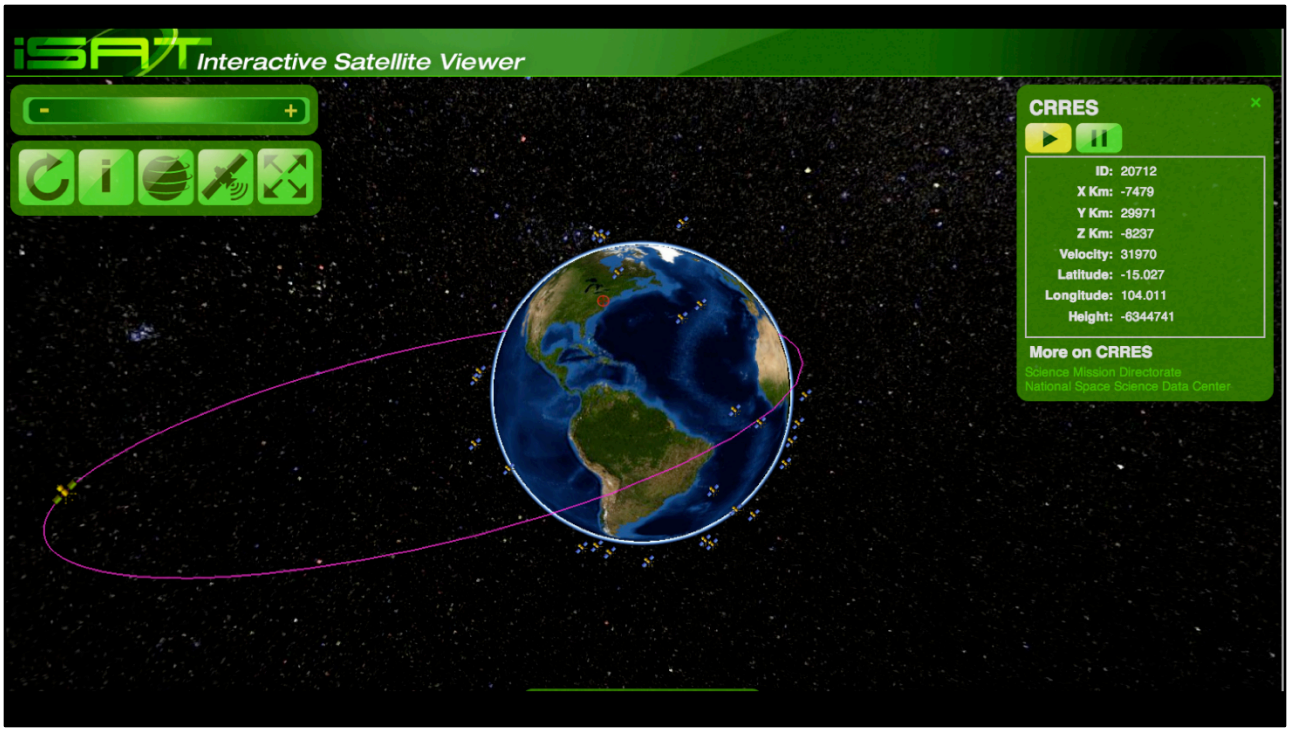
Browser-based calculations and 3D visualization.



Overview:

- Background and history
- How it's built, and why
- Demos in action

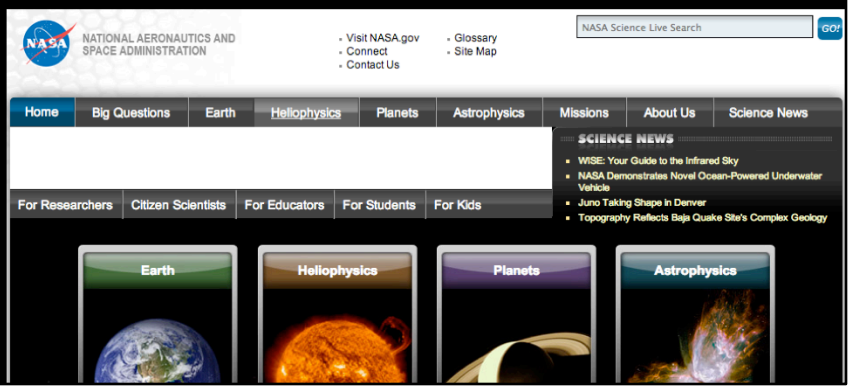
In the 1990s, Patrick Meyer at NASA MSFC was working on Fortran and Java code to track satellites.
1997: J-Track,
1998: J-Track 3D. It required a 166 MHz CPU
Implemented as a Java applet, run by the browser (not on the server)

http://www.nasasolutions.com/software-catalog/MFS-32013-1-J-Track.php

Science@NASA

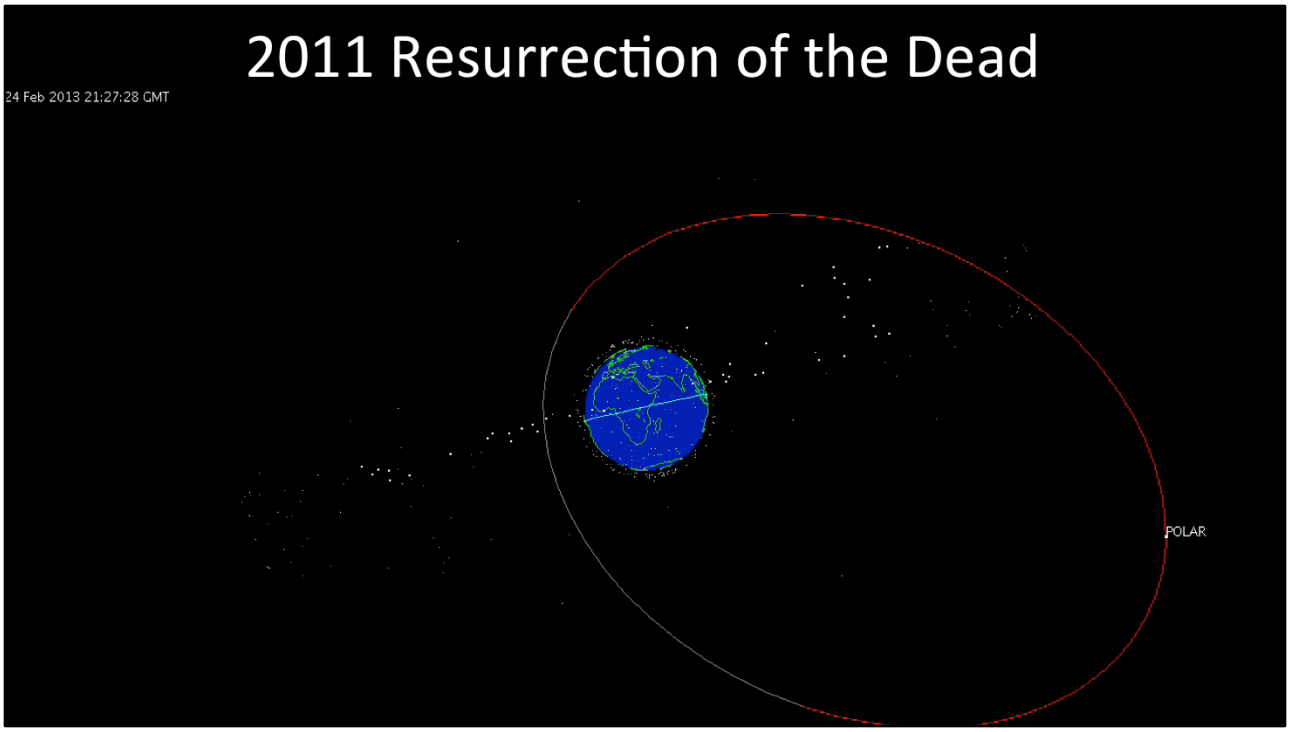
2010

NASAscience



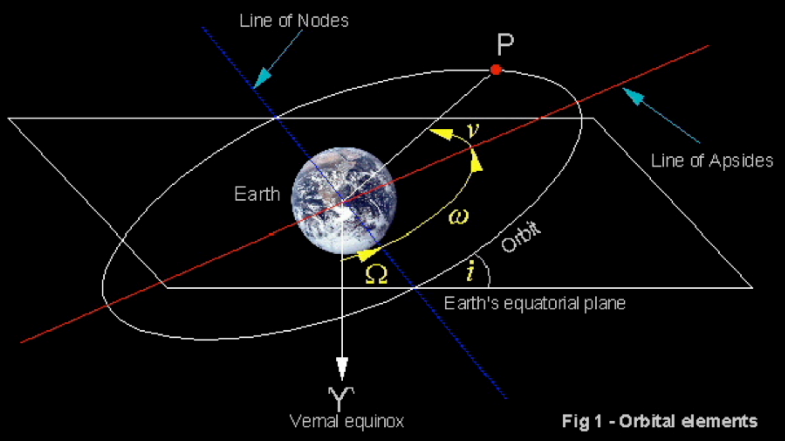
MSFC's Science@NASA which hosted J-Track 3D ran out of funding
April 2010 merged content into HQ's NASAscience site: over 10 years of stories.
We lost the applet's Java .class so J-Track 3D stopped.

This “Not Found” for J-Track was our biggest error in our web logs, by far.
We never realized how popular it was.
Found Java .class files on the Wayback Machine, decompiled it, got it working.
Tracked down Patrick Meyer and get the original code.



32second screen cast of J-Track 3D and the POLAR satellite.
100x Realtime.
Cool it's running again, runs fast on today's hardware.
Instead of improving the old Java code, we decided to implement using modern approaches.
Java on the client side is getting less and less useful.
Vendor support is waning, Security threats are increasing.
How can we re-implement it?

SGP model, TLE data




```
ISS (ZARYA)
1 25544U 98067A   13054.33198096  .00043022  00000-0  69361-3 0  2781
2 25544  51.6461 313.8194 0011796 301.9946 162.4257 15.52412156817027
```

SGPs predict the effect of the Earth, Sun and Moon on satellite orbits.
TLE examples: Red: name; blue: satellite number, green: international designator (YY +Number+Fragment); yellow: year; orange: day last sighted.
The rest are parameters of the orbit, for example, pink indicates revolutions per day.

Math's hard, let's go shopping!

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)

$$\theta_{GMST1982} = 67{,}310.548\ 41^s + (876{,}600^h + 8{,}640{,}184.812\ 866^s)T_{UT1} + 0.093\ 104T_{UT1}^2 - 6.2\text{x}10^{-6}T_{UT1}^3 \quad \text{(C-1)}$$

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).

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

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)

$$M_l = 134.962\ 981\ 39^s + 1{,}717{,}915{,}922.6330^s T_{TT} + 31.31T_{TT}^2 + 0.064T_{TT}^3$$
$$M_\odot = 357.527\ 723\ 33^s + 129{,}596{,}581.2240^s T_{TT} - 0.577T_{TT}^2 + 0.012T_{TT}^3$$
$$\mu_l = 93.271\ 910\ 28^s + 1{,}739{,}527{,}263.1370^s T_{TT} - 13.257T_{TT}^2 - 0.011T_{TT}^3 \quad \text{(C-3)}$$
$$D_\odot = 297.850\ 363\ 06^s + 1{,}602{,}961{,}601.3280^s T_{TT} - 6.891T_{TT}^2 + 0.019T_{TT}^3$$
$$\Omega_l = 125.044\ 522\ 22^s - 6{,}962{,}890.5390^s T_{TT} + 7.455T_{TT}^2 + 0.008T_{TT}^3$$

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

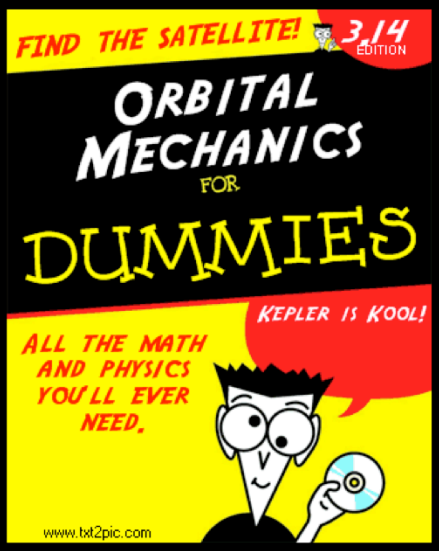
$$a_{p_i} = a_{i1}M_l + a_{i2}M_\odot + a_{i3}\mu_l + a_{i4}D_\odot + a_{i5}\Omega_l$$
$$\Delta\psi = \sum_{i=1}^{106}(A_{p_i} + A_{p_i}T_{TDB})\sin(a_{p_i}) \quad \Delta\varepsilon = \sum_{i=1}^{106}(A_{\varepsilon_i} + A_{\varepsilon_i}T_{TDB})\cos(a_{p_i}) \quad \text{(C-4)}$$

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)

$$\Delta\psi_{1980} = \Delta\psi + \delta\Delta\psi_{1980} \qquad \Delta\varepsilon_{1980} = \Delta\varepsilon + \delta\Delta\varepsilon_{1980}$$
$$\bar{\varepsilon} = 84{,}381.448^s - 46.8150T_{TT} - 0.000\ 597T_{TT}^2 + 0.001\ 813T_{TT}^3 \quad \text{(C-5)}$$
$$\varepsilon = \bar{\varepsilon} + \Delta\varepsilon_{1980}$$

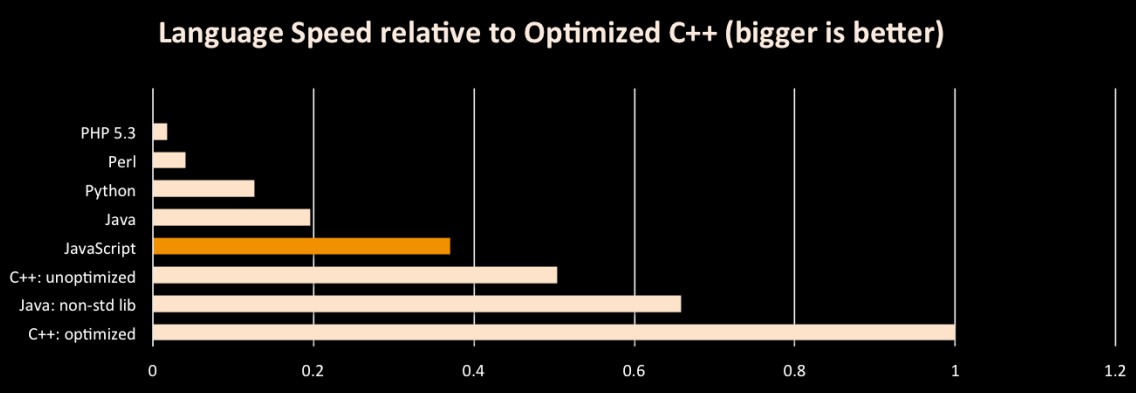
The equation of the equinoxes ($EQ_{opt1980}$) can then be found. The last two terms in the $EQ_{opt1980}$ are probably not included in AFSPC formulations. From McCarthy (1992:30)

$$EQ_{opt1980} = \Delta\psi_{1980}\cos(\bar{\varepsilon}) + 0.002\ 64^s\sin(\Omega_l) + 0.000\ 063\sin(2\Omega_l)$$
$$\theta_{GMST1982} = 67{,}310.54841^s + (876{,}600^h + 8{,}640{,}184.812\ 866^s)T_{UT1} + 0.093\ 104T_{UT1}^2 - 6.2\text{x}10^{-6}T_{UT1}^3 \quad \text{(C-6)}$$
$$\theta_{GAST1982} = \theta_{GMST1982} + EQ_{opt1980}$$



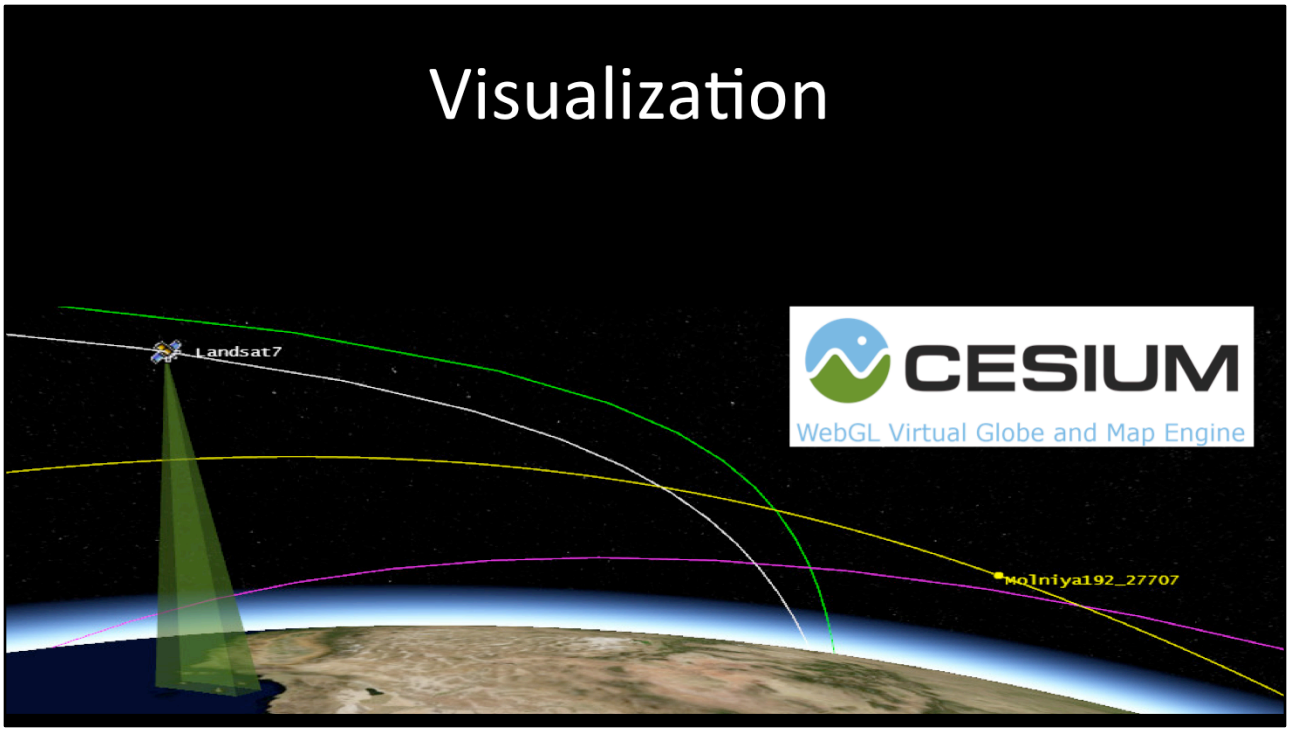
The math is hard.
The SGP code is dense.
We could do calculations on server and stream to browsers, but…
if we have 1000 satellites and a million browsers, we're going to crush the server.

JavaScript is Fast – Enough?



67,000 calculations/second !

Can we do it in the Browser itself?
JavaScript turns out to be pretty fast.
Graph based on
http://blog.famzah.net/2010/07/01/cpp-vs-python-vs-perl-vs-php-performance-benchmark/



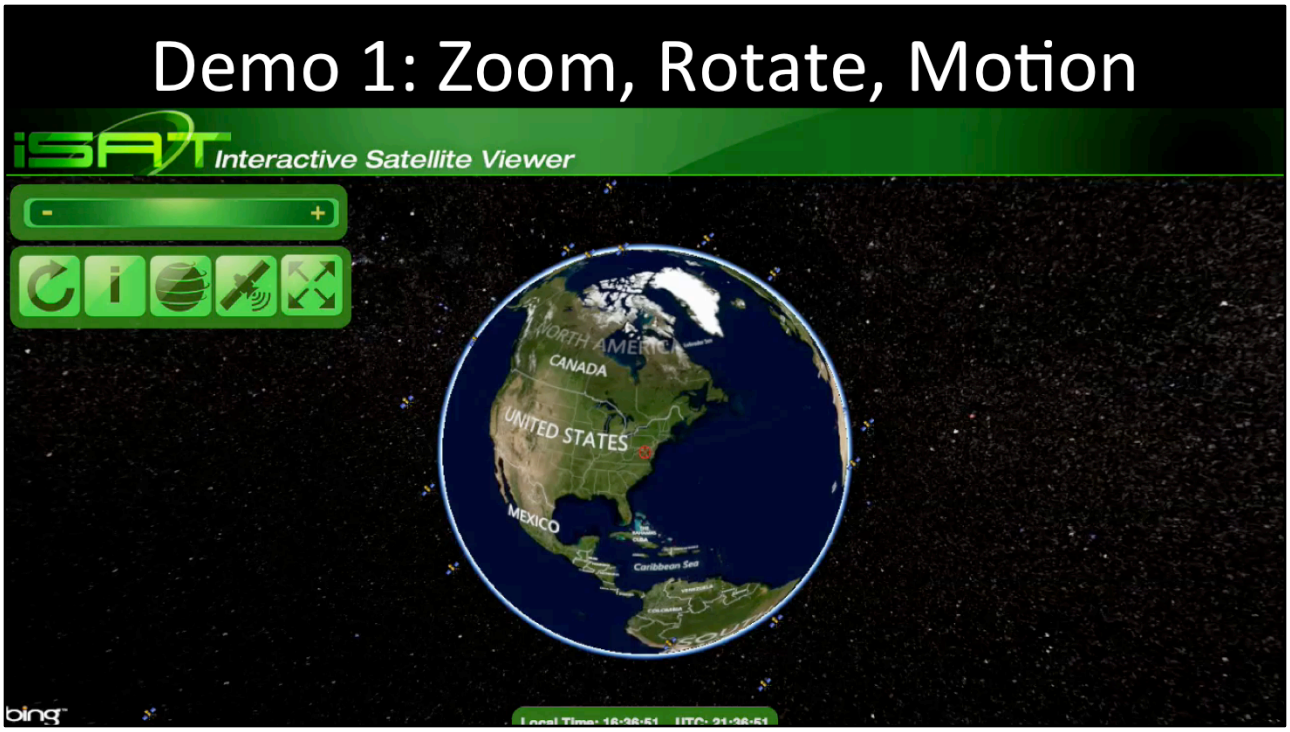
We have the positions, but how do we display them to users?
At last year's NASA Open Source Summit, I ran into some folks from AGI who told me about work they were doing in satellite visualization. I
All JavaScript, using WebGL, running in the browser – perfect.
It's open source, hosted in GitHub, and actively developed – currently at Beta 13.
Will it run in all browsers? Depends on WebGL support.
Support is improving – aim for the future rather than the past.

Demos

http://science.nasa.gov/iSat

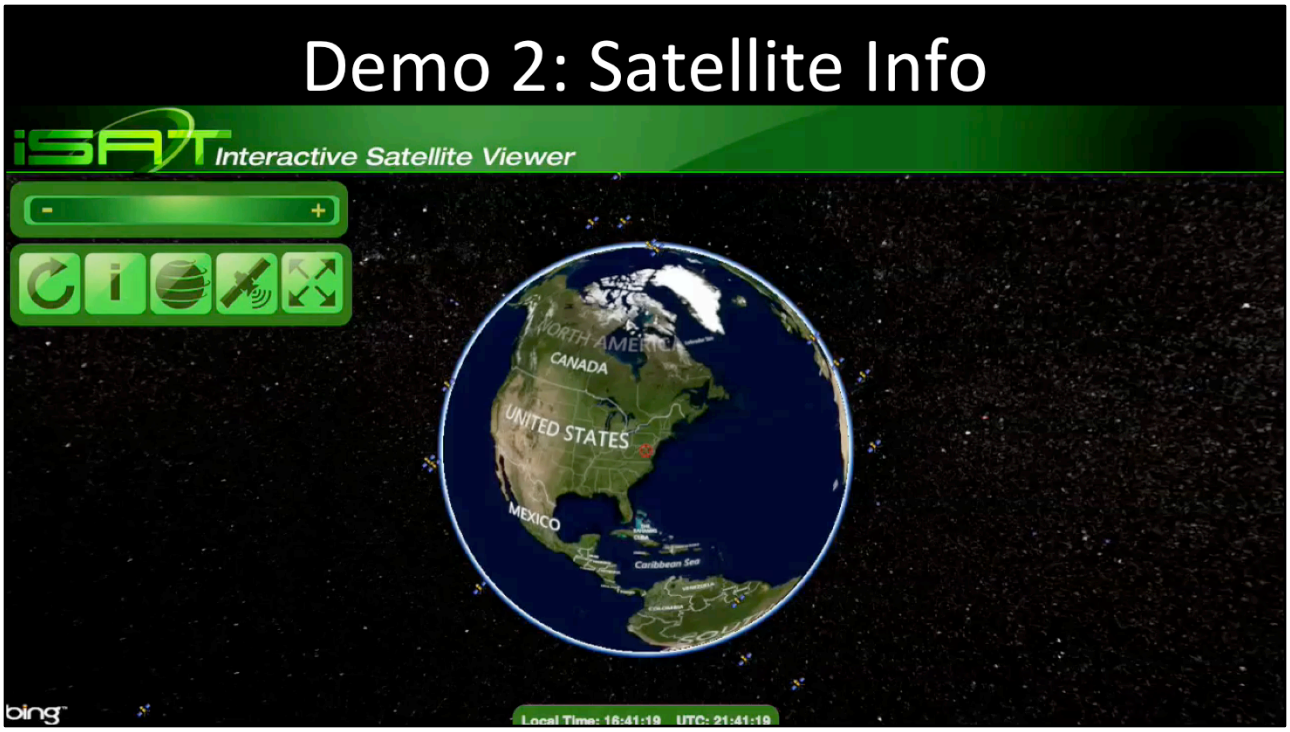
We're just using screencasts here to show some of the interactivity.
Try it yourself!
(QR code from phone: phone may not run app well, try it on a desktop)

Demo 1: Zoom, Rotate, Motion



Reset: fix orientation, centers on your geolocation.
Info: on basic usage.
Zoom out: show outlier satellites.
Rotate: show 3D with stars (looks great on full-screen)
Zoom in: show satellite flying over land in real time.

Demo 2: Satellite Info

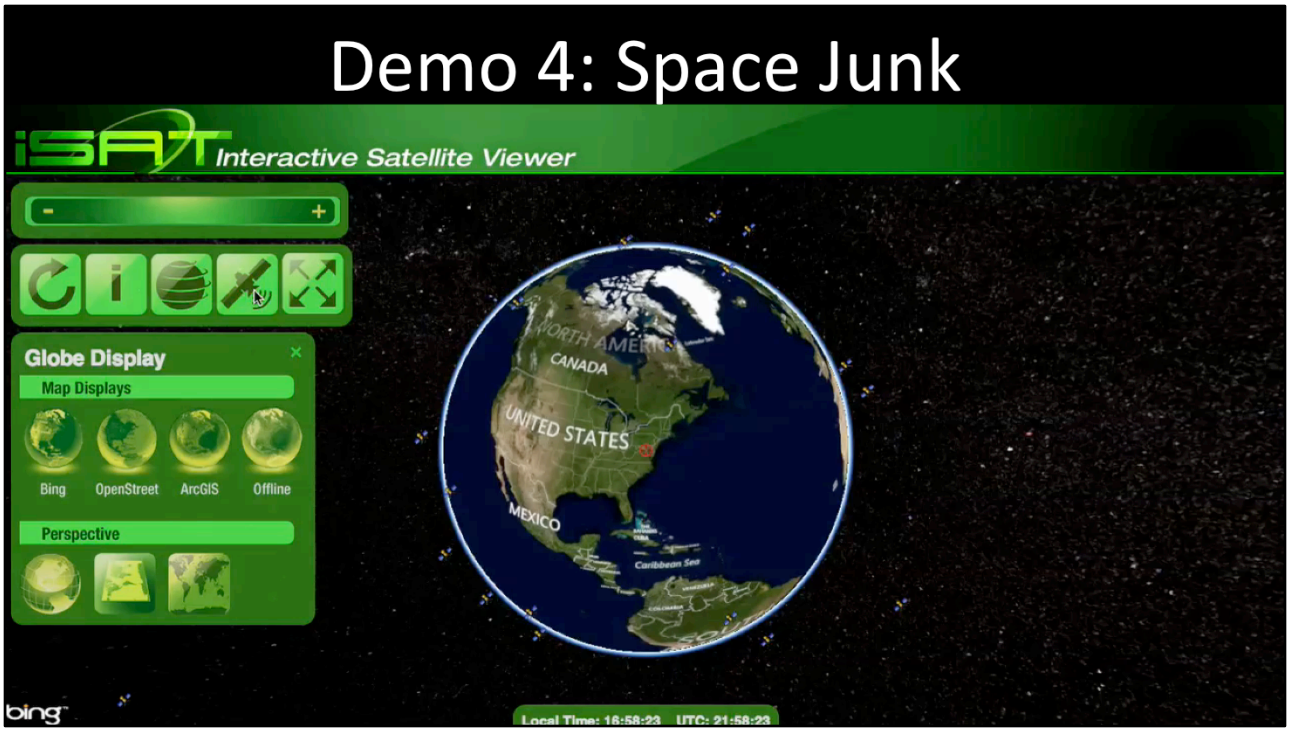


Hover: show name
Click: center on satellite; shows orbits, rotate to view.
Location pane; links to Science, NSSDC.

Demo 3: Globe, Satellite pickers

Map tile provider: Bing, OpenStreetMap, ESRI, flat file for offline; back to ESRI. (pick a satellite to get orbit)
Projection: spherical; flat; “2.5D” flat map with satellites above.
Satellite picker:
- NASA Science, POLAR
- Geosynchronous, same radius, on Equator, hole over ocean, dense over the Americas.

Demo 4: Space Junk



1997 Chinese test of satellite killer destroyed Fengyun-1C.
Over 2000 golf-ball sized pieces of debris, over 150,000 particles.
All moving in different directions.
Still in orbit, posing a hazard to others.

V1.0 Bugs and missing features
Bad location due to coordinate transformations.
Tighter integration with Science.nasa.gov and NSSDC.
Clock: x100, x1000 realtime
Better Android support in latest Cesium release.
Touch support for handheld devices.

- Team who helped build it
- Cesium community for virtual globe engine
- SMD and NASA for supporting the development
- SXSW for letting me debut it here

Play with it!

http://science.nasa.gov/iSat

@shentonfreude chris.shenton@nasa.gov