User Guide for Visualization Applet

Integration and Operation

Written by: Michael Mascaro

Summer 2011

**Table of Contents**

1. Applet Introduction.....................................................................................................page 2
2. Integration: HTML......................................................................................................page 2
3. Integration: JNLP........................................................................................................page 4
4. Integration: User Inputs...............................................................................................page 6
5. Integration: Ephemeris................................................................................................page 7
6. Operation: Getting Started..........................................................................................page x
7. Operation: Basic Functions.........................................................................................page x
8. Operation: Advanced Functions.................................................................................page x

**Applet Introduction**

The visualization applet has been designed as a replacement for the current visualization tools being used on the CCS website, bringing a more user-friendly, interactive experience while delivering faster results and more information than previous tools. It was coded in Java, using NASA World Wind software and code segments from an open-source satellite tracking program called JSatTrak. It can interactively display multiple satellites in orbit in both a three dimensional and a two dimensional view, using data read from ephemeris files. The scenario can be played at different speeds, or can even be viewed in a real-time mode. Since the applet is embedded on a web page, no download is required, and the applet can be configured to be running even before a user requests visualizations, and updates when the visualization is requested. This document will cover both the set-up and integration of the applet into the CCS website, as well as the operation and use of the applet from the user’s prospective.

**Integration: HTML**

The first step to integrating the applet into CCS is to place the files on a server where they can be accessed, and then embed the applet into an html page. The code for doing this is surprisingly simple. In HTML, the **<applet>** tag signifies the start of the applet, using the **ID** parameter to identify the name of the applet. Other parameters can specify the percentage width and height of the applet within the webpage. Java has developed two ways of deploying applets, one using purely html code, and another using a JNLP file to set up the applet. The HTML code can be developed, however, to use both. In the following code (a modified example of the HTML used for testing), the JNLP is tried first, and if it is not supported, the older method of loading applets is used.

<applet id="WWJApplet" mayscript code="org.jdesktop.applet.util.JNLPAppletLauncher" width=100% height=90%

archive="http://download.java.net/media/applet-launcher/applet-launcher.jar, worldwind.jar, gluegen-rt.jar, wwapp.jar, jogl.jar">

<param name="jnlp\_href" value="WWJApplet.jnlp">

<param name="codebase\_lookup" value="false">

<param name="subapplet.classname" value="WWJApplet">

<param name="subapplet.displayname" value="WWJ Applet">

<param name="noddraw.check" value="true">

<param name="progressbar" value="true">

<param name="jnlpNumExtensions" value="1">

<param name="jnlpExtension1" value="http://download.java.net/media/jogl/builds/archive/jsr-231-webstart-current/jogl.jnlp">

</applet>

The most important part of this HTML code is the first parameter, the “jnlp\_href.” This is where the HTML code links to the JNLP file, which sets up the applet. If this process does not work, then the applet is set up using the remaining HTML code. The format and usage of the JNLP file is discussed in the next section.

**Integration: JNLP**

The file that actually instructs the browser of the necessary files and setup for the applet is the JNLP file referenced in the HTML code for the webpage. This file, which must use the .jnlp file extension, is actually written in XML. In this file the locations of the required libraries for the applet must be detailed, as well as the security permissions of the applet and other information. Below is an example JNLP file, which is used in the local deployment of the applet for testing purposes.

<?xml version="1.0" encoding="UTF-8"?>

<!-- $Id -->

<jnlp href="WWJApplet.jnlp">

<information>

<title>CCS Visualization Test</title>

<vendor>a.i. Solutions</vendor>

<homepage href="http://ai-solutions.com"/>

<description>CCS Visualization Test Page</description>

<description kind="short">Visualization Test</description>

</information>

<security>

<all-permissions/>

</security>

<resources os="Windows">

<property name="sun.java2d.noddraw" value="true"/>

</resources>

<resources>

<j2se href="http://java.sun.com/products/autodl/j2se" version="1.6+" initial-heap-size="512m"

max-heap-size="512m"/>

<property name="sun.java2d.noddraw" value="true"/>

<jar href="wwapp.jar" main="true"/>

<jar href="worldwind.jar"/>

<jar href="jogl.jar"/>

<jar href="gluegen-rt.jar"/>

<jar href="plugin.jar"/>

<jar href="gdal.jar"/>

<extension name="jogl"

href="http://download.java.net/media/jogl/builds/archive/jsr-231-webstart-current/jogl.jnlp"/>

</resources>

<!-- Width and height are overwritten by the surrounding web page -->

<applet-desc

name="WWJApplet"

main-class="Main.WWJApplet"

width="800" height="600">

<param name="separate\_jvm" value="true" />

</applet-desc>

</jnlp>

The “all permissions” in the security tag means that the applet can have access to the computer system- vital for the applet being able to update to the system time. However, as a result of this every .jar library that is involved in the applet must be signed using the jarsigner tool in the Java Developers Kit. The same key must be used to sign each and every jar file, or else the applet will not run due to a security error. When listing the resources (all the libraries used by the applet), it is important to identify the library containing the applet itself by the value **main = “true”**. The reason for the value of the **main-class** in the **<applet-desc>** tag being “Main.WWJApplet” is because the main class, “WWJApplet,” is in the package “Main.”

**Integration: User Inputs**

Perhaps the most confusing part of the integration process is understanding the user input process of the applet and its code. The concept behind the user input was to create a file containing desired visualization requirements, and the scenario would then read this file and create the visualization as a result. Therefore, the format for the “parameters” file was created to match that of a compilation of desired visualization requirements. A corresponding java class was created to read this specific file format and translate that information into commands for the applet. The parameters format was originally designed to handle code for a model-centered view mode as well, but that functionality is currently not available in the applet. The categories that are included in the format are (in this order), “satellitename”, “ephemerislocation”, “get2Dsatcolor”, “plot2Dfootprint”, “viewobject”, and “scenariotime.”

* satellitename: This is for the name of the satellite(s)
* ephemerislocation: The direct link to the ephemeris (http:// included)
* get2Dsatcolor: The color of the satellite(s). Can be orange, yellow, red, white, blue, green, pink, or magenta
* viewobject: True or false whether the satellite(s) should be viewed (currently unavailable – should be false)
* scenariotime: The requested start time string of the scenario (must be within the earliest possible ephemeris point and the latest possible ephemeris point). The string must be in the following format: DD MMM YYYY hh:mm:ss

All values in this format should be lowercase, and there are no spaces in between the values. Each parameter has an equals sign following it, and then the value is listed without spaces. In the case of multiple satellites, each satellite must have information in every category (except for scenariotime), and the information for each satellite should be divided by a semicolon. An example of a scenario adding two satellites would have an input file that looked like this:

satellitename=Test 1;Test 2

ephemerislocation=http://myEphem.com/Ephemeris1.e;http://myEphem.com/Ephemeris2.e

get2Dsatcolor=orange;yellow

viewobject=false;false

scenariotime=15 Jul 2011 01:20:00.000

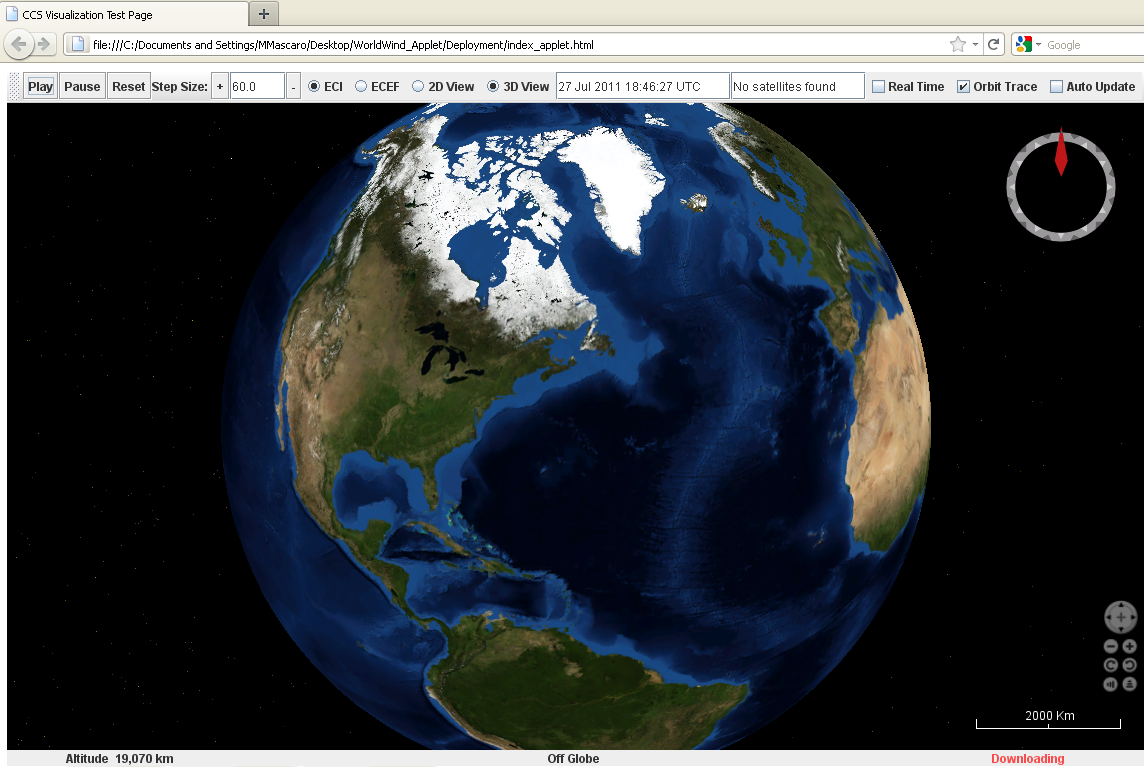
This input file would add two satellites, named “Test 1” and “Test 2” into the scenario, using the given ephemeris files. “Test 1” would be orange, and “Test 2” would be yellow. The scenario time would be the 15th of July, 2011, at 1:20 UTC, unless this time was not within the ephemeris ranges. It is hoped that this file could easily be created by a script on the server so that the user will see an interface of some sort for choosing their inputs, and the program will continue to see a file of this particular format to read and evaluate.

**Integration: Ephemeris**

The ephemeris files that are listed in the user input have to be direct links to ephemeris that use the STK format. The applet reads whatever happens to be at that URL, so the ephemeris cannot be text on a webpage, as the applet would attempt to read the source rather than the text visible to users. Currently, the functionality for reading more ephemeris file types is being developed, but has not been implemented, so the ephemeris must be STK format.

**Operation: Getting Started**

When the applet starts, regardless of user input or not, the interface will always be the same. The remaining part of this documentation details the description and use of the interface and how it pertains to the user.



The top part of the applet is the tool bar, where the majority of the user controls are found. Just below the tool bar on the right hand side of the applet is the compass, which helps orient the user when they are using the interactive features of the globe. The bottom right of the applet contains several grey buttons, which control the view of the 3D globe, which can also be controlled through the mouse. The bottom of the applet contains a status bar that identifies three things (from left to right), altitude of the camera, the latitude and longitude of the mouse, and the connectivity of the applet to the World Wind servers (for downloading more detailed images).

When viewing the applet, the first thing a user should know about is how to use the view controls. Clicking the mouse on any spot on the globe rotates the globe so that you are looking at that point. Clicking and dragging will also move the globe, as will using the large four-point button in the view controls (bottom right). The plus and minus buttons in the view control zoom in and zoom out, respectively, and can also be controlled by using the scroll wheel on your mouse (if applicable). The buttons in the view controls with spinning arrows (below the zoom buttons) roll the view in either direction, which can also be done by right clicking (holding) and moving the mouse left or right. The last view control buttons change the elevation of the view, from looking straight down at the Earth to looking at a ground-level view. This can also be done using the right mouse button (holding) and moving the mouse up and down.

**Operation: Basic Functions**

****

The left hand side of the tool bar contains all the basic commands and options for the applet, with the status displays located near the middle of the tool bar. The play and pause buttons are self-explanatory, they play and pause the animation of the scenario. The reset button resets the scenario back to the original time that had been requested by the user, and stops all animation that may have been in progress. The plus button next increases the step size by pre-decided amounts and the minus button lowers the step size by pre-decided amounts. In between these two buttons is the display for the current step size. The step size can also be customized by typing a new step size into the display and hitting enter. If the step size is in a correct format (a double or an integer), it will be entered into the scenario. If the step size is incorrect, it will display a message in the status display, and be ignored.

Next on the tool bar come the visualization options. First is a set of two radio buttons labeled ECI and ECEF. Clicking the ECI button (set as default) will make the 3D view an Earth-Centered Inertial view. Clicking the ECEF button will deselect the ECI button and make the view an Earth-Centered Earth Fixed view. Next to those buttons are the 2D and 3D view radio buttons. Clicking the 2D view button will change the scenario into a 2D view mode, and deselect the 3D button. Clicking the 3D view button (set as default) will deselect the 2D view button and change the scenario into 3D view mode. Next to these buttons are the displays for the current time of the scenario, and the status display. Neither of these can be changed, but the current time of the scenario will be automatically updated as the scenario animates, and the status display confirms the success or failure of every action done by the applet.

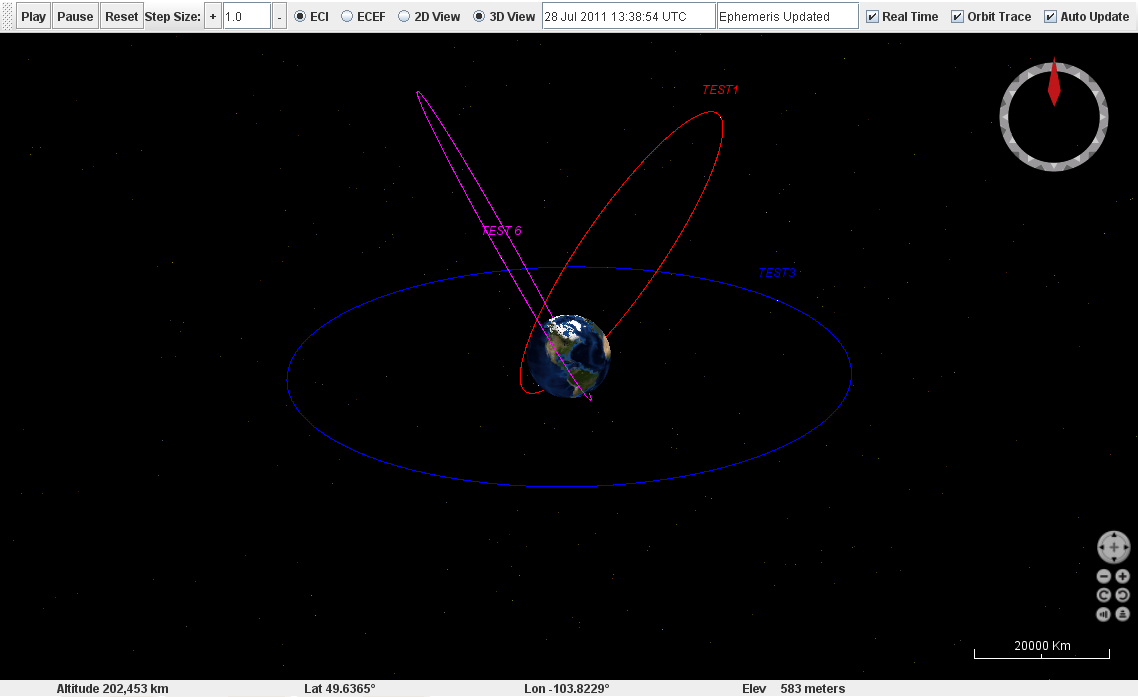
**Operation: Advanced Functions**



On the right hand side of the tool bar are the more advanced visualization functions that require a bit more explanation. Just after the status display are three checkboxes; the first checkbox is labeled “Real Time” and is unchecked as a default. Checking (clicking) this box will cause the scenario to switch into real time mode. In real time mode the step size becomes one second and the scenario is updated every second; in essence, the scenario will show the satellite orbiting as it is in that current moment. When the real time mode is selected, the scenario automatically starts playing, but unselecting real time mode will stop the scenario again. In addition, when the scenario goes to real time mode, obviously the scenario time changes; when real time mode is unselected, the scenario returns to whatever the scenario time had been before real time mode. For example, if the scenario is showing an orbit at 12:32 UTC and then real time mode is selected, the scenario will display the current orbit at that exact moment (for example 4:07 UTC). However, when real time mode is unselected, the scenario will return to 12:32 UTC. Since the real time mode still uses a timer within the program to iterate the scenario (1 second every second), it is also designed to be self correcting. Performance issues can cause the real time mode to be slightly slower than the actual time, and if run long enough the user may determine a difference of several seconds between the real time mode and the actual time. The applet is programmed to update to the system time if the difference between the system time and the scenario time becomes greater than 10 seconds.

The next option after the real time mode is labeled “Orbit Trace.” This checkbox (which as a default is selected) controls the visualization of the satellite(s) orbits. If selected, the visualization displays each satellite and their orbit in the 3D view, and displays each satellite and their ground track in the 2D view. However, if the view is becoming too crowded with multiple satellites and confusing orbits, the user can choose to deselect the “Orbit Trace” checkbox, and remove the orbits from the view. The satellites and their names will still remain visible, but the path of their orbits will no longer be visible. This is especially useful for visualizing multiple satellites in the 2D view.

The final checkbox in the tool bar is labeled “Auto Update” and is set unselected as a default. The auto update command, when selected, starts a second timer that automatically re-reads the user inputs into the scenario every six seconds. This means that if any of the satellite information is changed, it will appear into the scenario within six seconds of the change. Also, if the ephemeris files were changed or their locations changed, those corresponding changes will also appear in the scenario. The one aspect of this process that is unique from the initial loading of the scenario is the handling of the time inputs. If the scenario loaded correctly initially, then the scenario time is set to the time required in the user inputs, and once the auto update function has started, the scenario time is no longer changed by the updates. However, if the scenario initially loaded with an improper time, the update function will initially override the incorrect time with a correct one (if a corrected time is entered in the inputs), and then run as normal. Otherwise, the auto update function will ignore changing the time, allowing the user to run the scenario while the update is on. The reason for the auto update not changing the time is so the scenario won’t reset to the user requested time every six seconds, but continue playing as normal while still being able to adjust the ephemeris data or satellite information based on changes to the inputs.



A running scenario in real time mode with three satellites and auto update selected.