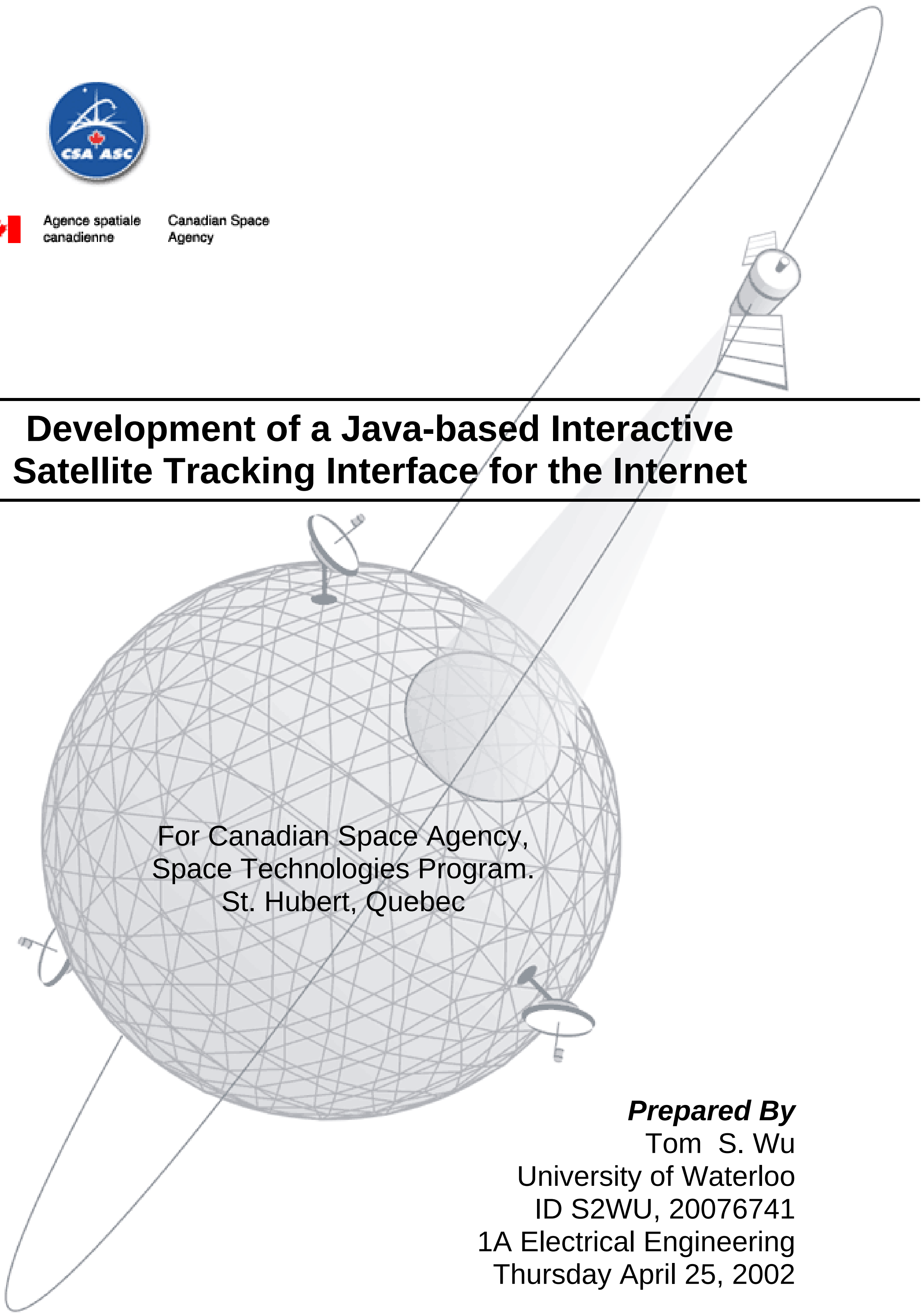
Agence spatiale canadienne Canadian Space Agency

Development of a Java-based Interactive Satellite Tracking Interface for the Internet



For Canadian Space Agency,
Space Technologies Program.
St. Hubert, Quebec

Prepared By
Tom S. Wu
University of Waterloo
ID S2WU, 20076741
1A Electrical Engineering
Thursday April 25, 2002

Beck Hall
108 Seagram Dr, Rm. 1104
Waterloo, ON, N2L 3B9

Thursday April 25, 2002

Dr. Anthony Vannelli
Chair E&CE
University Waterloo
Waterloo, Ontario, N2L 1G1

Dear Sir:

During my 1A work term, I prepared a report entitled "Development of a Java-based Interactive Satellite Tracking Interface for the Internet" for the Canadian Space Agency. The purpose of this report is to assess the feasibility of deploying a java-based satellite tracking interface on the agency's website.

I was employed on the groundstation software segment of the QuickSat project in the Space Technologies Division of the Canadian Space Agency. The goal of the Canadian Space Agency is to lead the development and application of space knowledge for the benefit of Canadians and humanity. The engineer in charge of my team was J.F Cusson, who is responsible for developing the ground control software used by the QuickSat micro-satellite.

This report was written at the request of Mr. Cusson. The contents of this report are entirely written by me and have not received any previous academic credit at this or any other institution. I would like to thank Mr. Cusson, for providing me with valuable advice relating to the structure of this report and proof-reading it. I have received no other assistance.

Sincerely,

Shu (Tom) Wu
ID 20076741

Contributions

During my employment at the Canadian Space Agency, I worked on the QuickSat project in the Space Technologies division. QuickSat is small team of around 20 engineers, scientists and students. The purpose of the project is to expand in-house expertise in spacecraft design and to develop a satellite for the flight demonstration of novel technologies developed by Canadian industries. My responsibility was to develop components for the satellite ground control software.

The team's goal is to create a relatively inexpensive micro-satellite similar and compatible to a class of satellites called the "amateur radio satellites". The satellite will allow the testing and demonstration of novel technologies developed by Canadian Industry. The satellite will have the advantage of being low-cost, light and modular. It will utilize off-the-shelf hardware and use existing ground infrastructure.

Another goal of the project is to develop in-house expertise in micro-satellites and to make the results available to Canadian Industry.

However, off the shelf products did not full fill the necessary requirements for software. The ground control software will be developed in house by computer engineer J.F. Cusson.

My duty as a software developer was to help Mr. Cusson develop and improve the ground control software. My first task was to develop a tracking interface that would allow users to visually see the orbital elements of a satellite. I also developed other graphical displays for the ground control software that takes raw telemetry data and presents them visually to the user. The tracking interface was my main project and will be the topic of my report.

Since I had no previous experience with Java, the development process also doubled as a learning process for me. This learning was done independently and did not obstruct the development process. With the exception of some design aspects, the design and development of the tracking Interface were completed by me. The client-server architecture of the Internet version was developed with the assistance of another co-op student, Louis-Philippe Ouellette.

As the tracking interface was being developed, it expanded beyond its original plan. More interactive features were added and in addition to being a part of the ground control software, it was also given the ability to run independently on a web browser. This version was deployed on the agency's intranet and was tested extensively.

The Internet version of the tracking interface went beyond the scope of the QuickSat team and is currently being reviewed by IT for possible deployment on the agency's website. This report focuses on the development of the tracking interface and the feasibility of deploying on the Internet.

Halfway though my term, my supervisor transferred to the spacecraft engineering team of the Space Technologies department. The ground control software is now being more generalized so that it will in the future be able to work with other satellite projects within the agency.

My work allows users of QuickSat's ground control software to visually see where various satellites are. Additionally, if the tracking interface is deployed on the Internet, it would offer an interactive, attractive and easy-to-use interface for Canadians and people around the world. By stimulating interest in Canadian Satellites and by making the experience fun, the Satellite tracking interface fits into the Canadian Space Agency's goal of extending the benefits of Space Technology to all Canadians.

Summary

The purpose of this report is to document the development of a Satellite-tracking Interface for the Quicksat’s ground control software and the Internet. This report uses a minimum of technical terms; however readers should have a basic understanding of Object-Oriented programming.

The report shows the development of the java object structure, interactive features, and the website. Important interactive features include zoom, satellite selection and time interaction. It also covers many technical challenges such as making reliable and accurate orbital predictions.

The project was concluded a success. Tests included trial runs on the local intranet. The accuracy was proven by passes with the International Space Station, which resulted in signal acquisition by the ground station. The program was shown to be easy to use, fast, reliable and visually attractive.

The recommendation is for the tracking interface to be reviewed by the IT department of the Canadian Space Agency for possible placement on the agency's website.

Conclusions

From the analysis in the report body, it was concluded that the development of the Satellite tracking Interface, called SatMapper, was a success and that it should be deployed on the Agency Website.

The above conclusion was determined through the analysis and testing of the application features. Additionally, the interactive and user-friendly nature of SatMapper makes it ideal for deployment on the Agency's website. The report documents the creation of a server-client architecture designed specifically for the Internet version.

Many technical challenges were overcome through the development process. SatMapper was successfully integrated into two separate environments, the Internet and the QuickSat Groundstation software. Many of the new features were conceived and added during the development process. All features were tested with the creation of a server and website on the agency's Intranet. Team collaboration was a problem at first but was solved through the code sharing software called WinCVS.

Accurate orbit prediction was accomplished by the use SGP4 formulas and up-to-date Keplerian data from the NORAD web site. For the Internet version, the orbit prediction of satellites is calculated on a server to insure accuracy because clients may not have an accurate clock.

The Satellite interface was able to successfully implement a dynamic interface features such as zoom using a 2-level coordinate system. Other features such as satellite selection and map transitions also employ animations for visual enhancement. Users are able to select satellites and interact with paths and time. Some advanced features such as interface control are accessible through an options dialog.

Recommendations

In order to address the possible deployment of SatMapper on the agency’s website, the program was passed on for review by the agency’s IT department. It should be considered for possible placement on the Agency’s website as an educational tool.

In the future, the website for SatMapper could be developed further to make it an effective educational tool. It could include links to Canadian space-related sites and information regarding Canadian satellites.

From a technical standpoint, it will be easy to add more choice of satellites. In addition to Canadian satellites, SatMapper could be used to track weather, science, amateur and other satellites of interest. Adding this feature will allow make the application more flexible and useful.

Table of Contents

List of Figures

List of Tables

1 – Introduction

The purpose of this report is to document the development process of SatMapper, an interface for visualizing the orbital parameters of satellites. This program was made at the request of Jean-Francois Cusson, the engineer in charge of QuickSat's Groundstation software.

The original plan by Mr. Cusson was the development of a satellite-tracking panel for QuickSat's Groundstation software. The requirement was that a simple interface be made through which the orbital parameters of a satellite can be visually presented to the user.

As the program developed, many innovative features were added that expanded beyond the original plan. SatMapper developed a highly interactive and functional interface. Combined with the use of high-quality maps, it offered visually attractive solution that was superior to others.

After SatMapper completed its original mandate, it was decided that it be developed as a separate entity. Since SatMapper was developed using Java, it could run directly inside a user's browser. This made it ideal as an informational tool that could be deployed on the Canadian Space Agency's website. It will promote interest among Canadians for the space program and the work of Space Technologies.

This report will highlight some important features of SatMapper and also addresses the feasibility of using it on the Internet. Since SatMapper is still pending approval, this report can be used as a proposal for deploying SatMapper on the Canadian Space Agency website.

1.1 – Background

The Space Technologies department of the Canadian Space Agency is designing and building the engineering model of a micro-satellite and its ground operations infrastructures. This satellite, named QuickSat, is intended for flight demonstration of novel technologies developed by Canadian industries while establishing flight heritage for their new technologies. The project also allows the Agency to expand its in-house technical expertise in spacecraft design.

The current design calls for QuickSat to be similar to and compatible with a class of satellites called the "amateur radio satellites". One of the advantages of this approach is to allow the testing of the ground infrastructure with existing satellites. Another one is the low cost (for the satellite as well as the ground infrastructure) of the project as compared with a more conventional project.

As such, the ground control system is using the Agency's amateur radio installations and hardware. This includes UHF/VHF radios, terminal node controllers, computers, antennas and a computer controlled antenna rotator.

For the software, however, off the shelf products were not fulfilling the requirements and in-house development is underway. The QuickSat's Groundstation software is being developed in Java. At the beginning of this term it included modules for:

- Controlling the radio station hardware, including the radios, terminal node controllers and antenna orientation;
- Communicating with the satellite in its native protocols;
- Controlling the satellite's hardware and software functionality;
- Getting and interpreting telemetry from the satellite;
- Predicting satellite orbits, to know when signal should be acquired, control the antenna rotation and control the radio to compensate for the signal frequency shift due to the Doppler effect; and
- Managing a small satellite database, to help the user switch between various satellites.
- Allowing the remote operation of the control station using conventional network links.

Two students were hired during the winter of 2002 to implement the following additional modules:

- A video streaming system to allow for remote/real-time visualization of the antenna via an Internet connection;
- A software panel designed for visualizing the result of the satellite orbit prediction module, in an intuitive and useful way; and
- A system to synchronize the application time with the official universal time (UTC), independently from the host computer clock.

At the beginning of the term, the groundstation software presented orbital parameters simply in a column of numbers. This report focuses on the development of a software panel that takes this column of numbers and performs all the necessary processing and transformations so that the user is able to see a visual representation of this data.

This software panel was highly successful and a secondary goal was added. The software panel became SatMapper, a java applet that includes the orbit prediction module and is capable of autonomous update of the satellite's orbital elements. And it was proposed that it be installed on the Agency's official web site as a promotional and educational tool.

1.2 – Scope

This report is written for J.F Cusson, the computer engineer in charge of QuickSat's Groundstation software. However, a minimum of technical terms was used in order to make this report accessible to other member's of the Canadian Space Agency. Since SatMapper was developed using Java, this report assumes readers have some basic understanding of object-oriented programming.

To simplify the understanding of the interface, many screenshots are provided to show visual features. Additionally, diagrams are provided to show object-oriented design and to simplify understanding of the interface interactions.

1.3 – Outline

This report will attempt to cover as much as possible on the development of SatMapper. However, to make this report less technical and to condense it, some parts of development have been avoided.

The report will begin with a summary of technical challenges and then a description of the development environment in Sections 2 and 3. The next section, Section 4, Determining Satellite Telemetry, will summarize the basic premise for making orbital predictions. However, the details of the calculations will be omitted because they are beyond the scope of the report. Java Object Structure, Section 5, will show the structures of the two versions of SatMapper through diagrams. All the interactive features are explained with the aid of figures and screenshots in Interface Interactions, Section 6. Next, Section 7 will show how SatMapper is able create a dynamical interface through the use of two coordinate systems. Website development is covered in Section 8 and Testing in Section 9.

1.4 – Advantages for use on the Internet

SatMapper was originally conceived as visual tracking panel as part of QuickSat's Groundstation software. However, as it evolved and more features were added, it was discovered that SatMapper could function as an independent application. Because it is written in JAVA, it is able to run on any platform whether it is Windows, Macintosh, UNIX, or Linux. People with sun's java-plugin will be able to view SatMapper directly in their web browsers. This made SatMapper ideal for deployment on the Internet.

As of now, the Canadian Space Agency's website only allows the public to view the locations of Satellites through NASA's J-Track [**1**]. SatMapper would provide a Canadian perspective because it is developed in house. Table shows the advantages SatMapper has towards J-Track:

Table 1. Feature Comparison between SatMapper and J-Track

J-Track [1] (currently used by the CSA)	SatMapper
English Only	Bilingual
Shows satellites of American interest	Shows satellites of Canadian interest
Accuracy dependent on accuracy of local clock	Server-based model allows clock to be synchronized
Limited interactivity	Interactive features allows user to zoom, move around the world. Users can select satellites for more information and control what is seen on the screen
No time interaction	Advanced time interaction allows users to go into the future and control the speed of time
Unintuitive interface that is slow and uses the keyboard	Uses intuitive interface that is completely mouse controlled and fast
Political map that is visually displeasing	Uses visually attractive and textured maps of the world. Allows users to interactively select maps of interest such as night and day.
No Animations	Visual features such as animation, transparency make satellite tracking visually attractive
Not able to show places	Able to show places, such as major Canadian Cities
Only available in Java	Available in static image format so people without Java can view it

SatMapper has been tested extensively within CSA's local intranet and has gone through multiple test phases. During development, a server was created on the Intranet to test the client-server model. SatMapper has been shown to be accurate, fast, and reliable.

SatMapper would add valuable tool to the Canadian Space Agency's website. It would offer bilingual Canadian content through an easy-use interactive interface. Because of its visual nature, it will be accessible to all Canadians, even those with no background in space science. It could be used as a promotional tool for space technologies and its other projects.

2 – Technical Challenges

Here is a summary of technical challenges that were identified at the beginning of the project and a their solutions.

- **Prediction algorithm had not been tested**
  The satellite prediction algorithm made by J.F. Cusson had never been used on multiple satellites at once. However, testing with up to 50 satellites proved that there were no problems.

- **Team Collaboration**
  Since the members of the QuickSat team were located in separate rooms of the CSA building, collaboration was a problem at first. This was solved using CSA internal messaging system and WinCVS software **[3]**. Team members could work on the same file at the same time from different locations and then compare changes.

- **Modular Design**
  SatMapper had to be able to function both as a standalone web application and as part of QuickSat's Groundstation software, JLoad. In the end, one SatMapper was able to perform both functions. The program was able to adapt to the environment in which it is being used.

- **Accurate Orbit Prediction**
  There was concern that in the web version of SatMapper, clients would not be able to accurately track satellites because their computer clock is inaccurate. This problem is negated by the use of a server, which sends orbital data to the client.

- **Finding Appropriate Maps**
  It was a problem at first finding high-quality and high-resolution textured maps. It was soon solved by NASA's Visible Earth web site, which provided the maps needed. The maps are public domain and the copyright policy can be found in Appendix A.

- **Optimization of Speed**
  Since one goal of SatMapper is to be as accessible as possible, it should run on most computers. SatMapper's code was optimized to be efficient and usable even on slow computers.

  On a Pentium 400, it was tested that SatMapper used on average about 17% of the CPU when animations weren't running. This was accomplished by the use of threads and by making the code intelligent. Complex operations are performed only when necessary and animations run in threads that do not interfere with normal operation.

  Additionally, the server will generate a static image version of SatMapper so that even people without Java will be able to view it.

- **New Untested Features**
  There were many special features that had never been tested before. It was not certain whether accuracy and reliability would be compromised by the complexity of some new features.

  The object-oriented nature of Java made the implementation of such features efficient and reliable. A network-server to host the SatMapper applet was created to allow the testing of SatMapper from any computer within the CSA Intranet.

- **Size of Program**
  There were concerns that SatMapper program will be excessively large in size for deployment on the web. However, after selectively including only relevant code, SatMapper code was reduced to 77K in size. And the low-resolution version of SatMapper will be around 0.25MB in size with images.

3 – Development Environment

This application was developed for the Java 2 Platform, Standard Edition 1.4, using Java Forte™ community edition v3.0. The software used was free and publicly distributed by Sun for non-commercial use. The operating system used was Microsoft Windows 2000. However, the application is platform-independent, meaning it could run on any platform.

No other third-party tools, components or code was used with the exception of the JPEG encoding algorithm. JPEG (Joint Photographic Experts Group) is an image compression algorithm commonly used on the web. This algorithm **[3]** was obtained from a third party and allowed the creation of a static-image version of SatMapper.

The WinCVS software assisted collaboration with other members of the team. WinCVS creates a code repository, which allows multiple members of the team to view and modify code from different locations. This aided the collaboration with the other members of the team who were in separate parts of the building.

The creation of the SatMapper homepage was developed using CorelDraw, Adobe Photoshop, and Macromedia Dreamweaver.

The application was developed using Java 2 technology. This requires the users to have a relevant Java 2 plugin in order to run it. The plugin is available for free on Sun Microsystems's website and is available for many operating systems.

4 – Determining the Orbital Parameters of a Satellite

The orbital parameters of a satellite are calculated in an object provided by J.F. Cusson. This object is responsible for providing coordinates, altitude and velocity of satellites. These calculations are performed for all Satellites at controlled intervals to animate the movement of satellites.

These calculations are based on Keplerian data retrieved from the NORAD web site **[4]**. This data is updated daily because satellite orbits change slightly over time. The Keplerian data contains all the elements required to accurately predict satellite coordinates within a few days. Figure 1, on page 7, shows the way Keplerian data is organized and how orbital parameters are calculated.

There are many mathematical models that can be applied on the Keplerian data. Since SatMapper is mainly used to track low-orbit satellites, the SGP4 model **[5]** was selected. The SGP4 model was developed by Ken Cranford in 1970 and is ideal for satellites with periods lower than 225 minutes. For satellites with longer periods, the formulas will lose their accuracy.

The formulas used in SGP4 can be found in Appendix B. These pages were taken from Space Track report No. 3. It is considered a standard and is used widely used in the amateur satellites community. The explanation of these formulas is beyond the scope of this report, so it will not be covered.

Format of Keplerian Data [6]

Line 1

Line 1 is a eleven-character name.

Line 2		*Line 3*	
01-01	Line Number of Element Data	03-07	Satellite Number
03-07	Satellite Number	09-16	Inclination [Degrees]
10-11	International Designator (Last two digits of launch year)	18-25	Right Ascension of the Ascending Node [Degrees]
12-14	International Designator (Launch number of the year)	27-33	Eccentricity (decimal point assumed)
15-17	International Designator (Piece of launch)	35-42	Argument of Perigee [Degrees]
19-20	Epoch Year (Last two digits of year)	44-51	Mean Anomaly [Degrees]
21-32	Epoch (Day number and fractional portion of the day)	53-63	Mean Motion [Revs per day]
34-43	First Time Derivative of the Mean Motion divided by 2. Or Ballistic Coefficient (Depending of ephemeris type)	64-68	Revolution number at epoch [Revs]
45-52	Second Time Derivative of Mean Motion divided by 6. (Blank if N/A)	69-69	Check Sum (Modulo 10)
54-61	BSTAR drag term if GP4 general perturbation theory was used. Otherwise, radiation pressure coefficient.		
63-63	Ephemeris type		
65-68	Element number		
69-69	Check Sum (Modulo 10)		

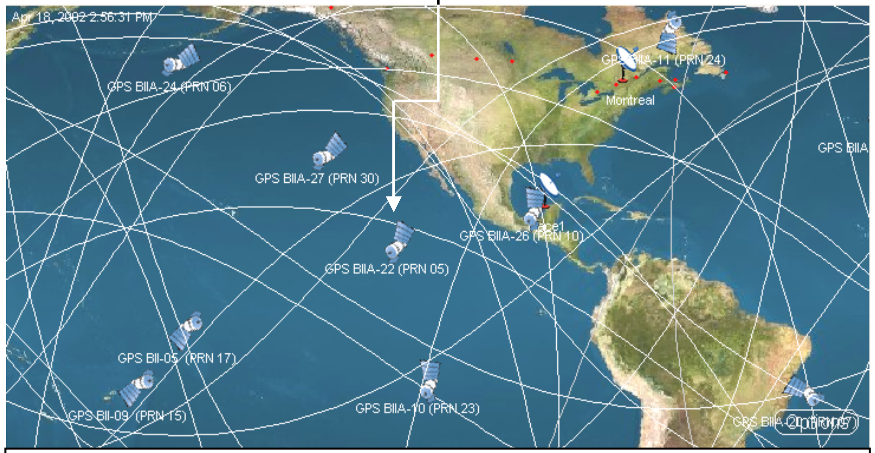
Keplerian Data from NORAD

GPS BIIA-22 (PRN 05)

```
1 22779U 93054A   02105.57229159  .00000027  00000-0  00000-0 0  7904
2 22779  53.6082 199.6641 0035060  29.1491 331.0960  2.00553809 63192
```

UTC Time
Time is used to determine satellite positions.

Calculations
Formulas are based on SGP 4 and can be found in *Appendix B*. These calculations are encapsulated in an object called Satellite (*Figure 3)*. Calculations provide accurate satellite coordinates.



A View of GPS Satellites Calculated from Keplerian Elements

Figure 1. Calculation of Keplerian Data

5 – Java Object Structure

SatMapper was built in an object oriented programming language called Java. Large functions and processes are encapsulated in what are called objects, which are easily transferable from one application to another. SatMapper itself is an object, which means it can be easily ported to applications.

This report focuses on SatMapper's use mainly as a standalone application for the Internet. However, SatMapper's original intent was to be used as a panel for QuickSat's groundstation software. The object structure for SatMapper's integration into JLoad is shown in Figure 2 (page 9).

5.1 – Server-Client architecture

The version of SatMapper for the Internet uses a client-server structure, shown in Figure 3 (page 10). Calculating the orbital parameters of the satellites on the server insures accuracy because the client's computer may not have an accurate clock. However, if the server is down, SatMapper will automatically perform calculations on the local machine and alert the user.

The real-time calculation of accurate orbital parameters depends heavily on the accuracy of the computer's clock. In order to ensure the accuracy of the data received, it was decided a server with accurate time would be responsible for sending the satellite coordinates at set intervals to clients.

Another co-op student, Louis-Philippe Ouellette, conceived the server-client model (appendix A). The implementation of his model into SatMapper was done in collaboration with him.

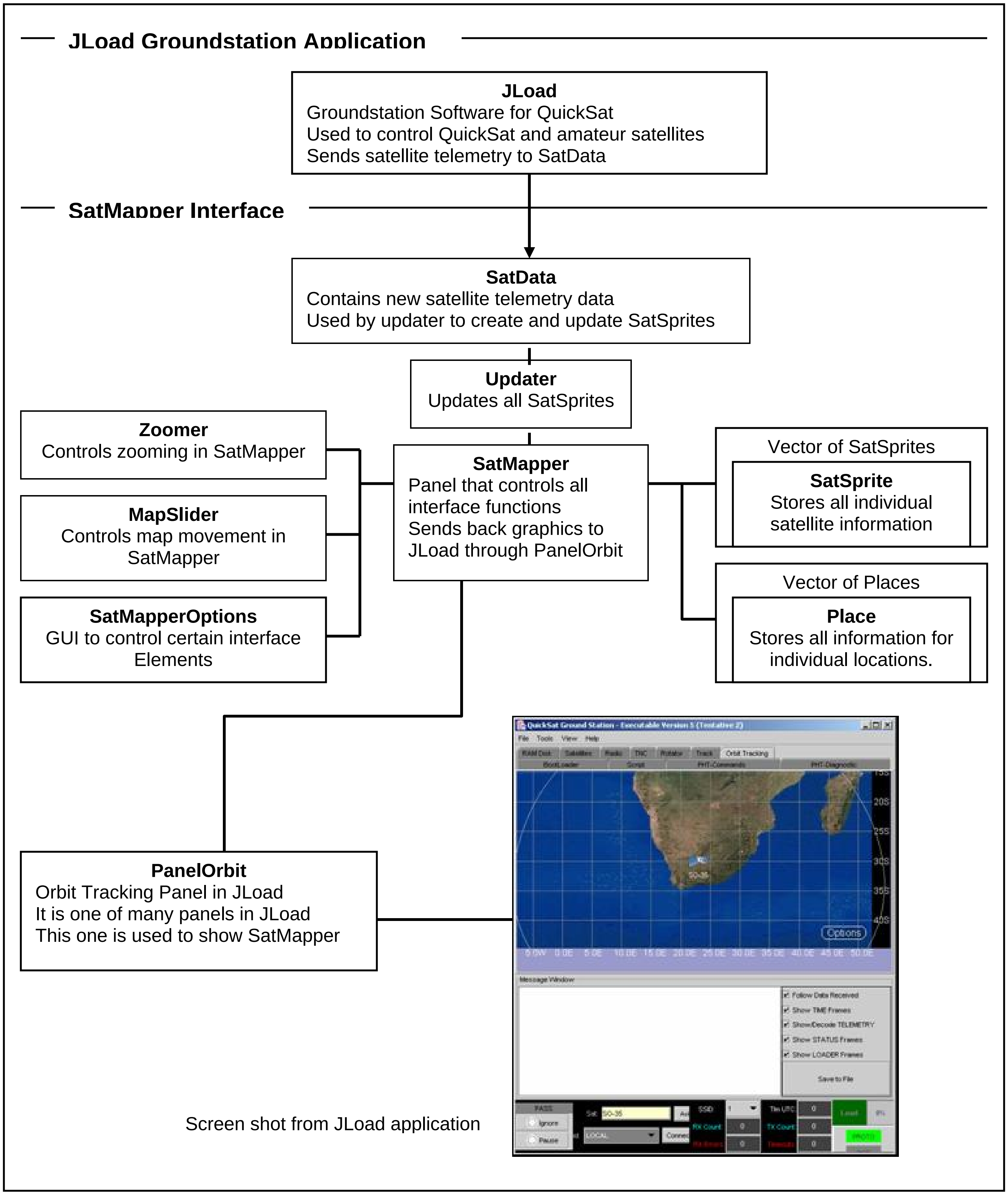

JLoad Groundstation Application
JLoad
Groundstation Software for QuickSat
Used to control QuickSat and amateur satellites
Sends satellite telemetry to SatData
SatMapper Interface
SatData
Contains new satellite telemetry data
Used by updater to create and update SatSprites
Updater
Updates all SatSprites
Zoomer
Controls zooming in SatMapper
MapSlider
Controls map movement in SatMapper
SatMapperOptions
GUI to control certain interface Elements
SatMapper
Panel that controls all interface functions
Sends back graphics to JLoad through PanelOrbit
Vector of SatSprites
SatSprite
Stores all individual satellite information
Vector of Places
Place
Stores all information for individual locations.
PanelOrbit
Orbit Tracking Panel in JLoad
It is one of many panels in JLoad
This one is used to show SatMapper
Screen shot from JLoad application

Figure 2. SatMapper object structure for JLoad groundstation software

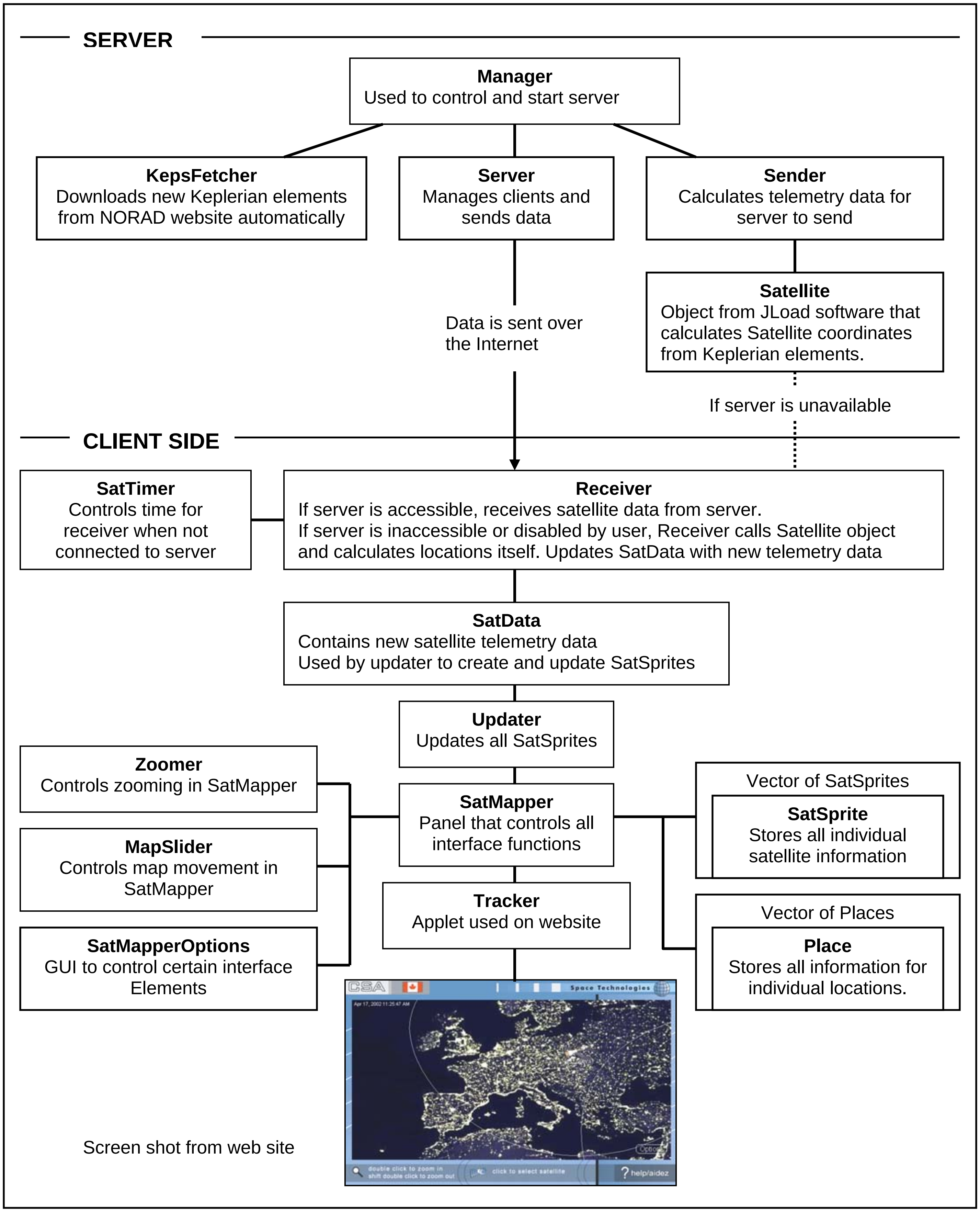

SERVER
Manager
Used to control and start server
KepsFetcher
Downloads new Keplerian elements from NORAD website automatically
Server
Manages clients and sends data
Sender
Calculates telemetry data for server to send
Satellite
Object from JLoad software that calculates Satellite coordinates from Keplerian elements.
Data is sent over the Internet
If server is unavailable
CLIENT SIDE
SatTimer
Controls time for receiver when not connected to server
Receiver
If server is accessible, receives satellite data from server.
If server is inaccessible or disabled by user, Receiver calls Satellite object and calculates locations itself. Updates SatData with new telemetry data
SatData
Contains new satellite telemetry data
Used by updater to create and update SatSprites
Updater
Updates all SatSprites
Zoomer
Controls zooming in SatMapper
MapSlider
Controls map movement in SatMapper
SatMapperOptions
GUI to control certain interface Elements
SatMapper
Panel that controls all interface functions
Tracker
Applet used on website
Vector of SatSprites
SatSprite
Stores all individual satellite information
Vector of Places
Place
Stores all information for individual locations.
Screen shot from web site
CSA
Space Technologies
double click to zoom in
shift double click to zoom out
click to select satellite
help/aidez

Figure 3. SatMapper client-server object structure for the Internet

5.2 –Satellite and Place Objects

Places and satellites are two of the most important objects in SatMapper. They represent actual physical things on the Geographic Plane. Place objects are locations of interest such as groundstations and major cities. Satellite objects, called SatSprites, represent satellites in space.

Because there could be multiple satellites and places, these two objects are stored in their two respective Vectors. Vectors are containers that can hold multiple objects.

These two types of objects interact with one another as they do in the real world. When satellites come within line of sight of a place, this contact is visually shown to the user (Figure 4).

Satellite objects, called SatSprites, contain all the information relevant to that satellite, including the coordinates of the path and line of sight circles. A SatSprite object is selectable, meaning the user could click on a satellite to focus more attention on it; this is discussed in detail in the Interactive Features section of this report.

Place objects store coordinates and name of an important location. They serve as reference points to groundstations and places of interest. Users can add their own home to the map using the options menu.

The distance between the coordinates of the satellite and place are calculated using the following formula.

lat1 = the latitude of the satellite	*lat2 = the latitude of the place*
long1 = the longitude of the satellite	*long2 = the longitude of the place*

$$Distance = radius * ACos[\sin(lat1) \sin(lat2) + \cos(lat1) \cos(lat2) \cos(long1-long2)] \quad \textbf{(1) [7]}$$

Using the distance formula **(1*)***, SatMapper cycles through all the places and checks if any Satellites are within range.

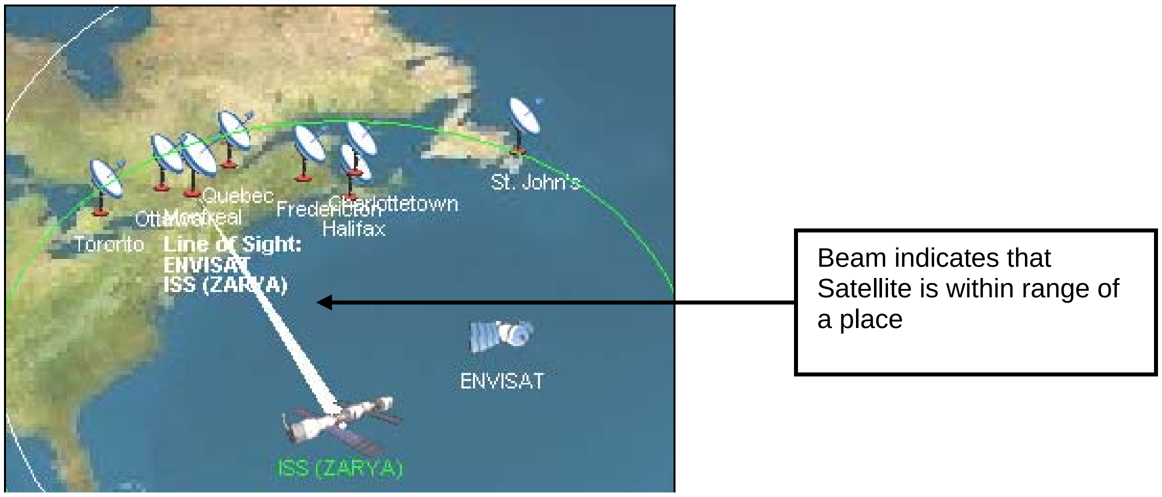


Figure 4. Beam indicating Line of Sight

5.3 Directory Hierarchy

Figure 5 shows the directory structure of all the java source files necessary to run the Internet version of SatMapper. To explain all of them would be beyond the scope of this report so they will simply be listed for reference here. The files are listed in alphabetical order.

The contents of some important files are located in Appendix C. However, because the length of the document is extremely long, it may not be included with this report.

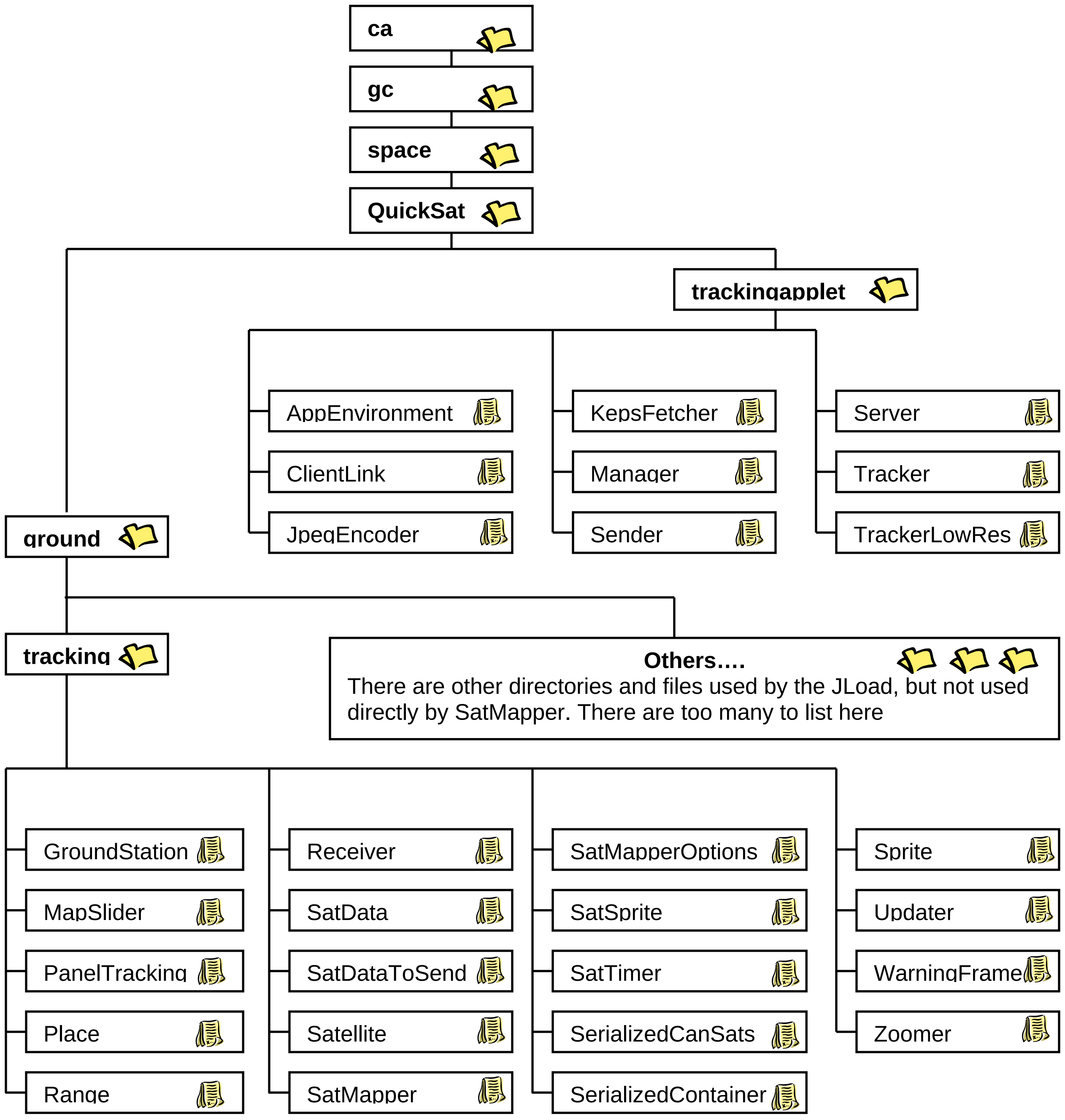
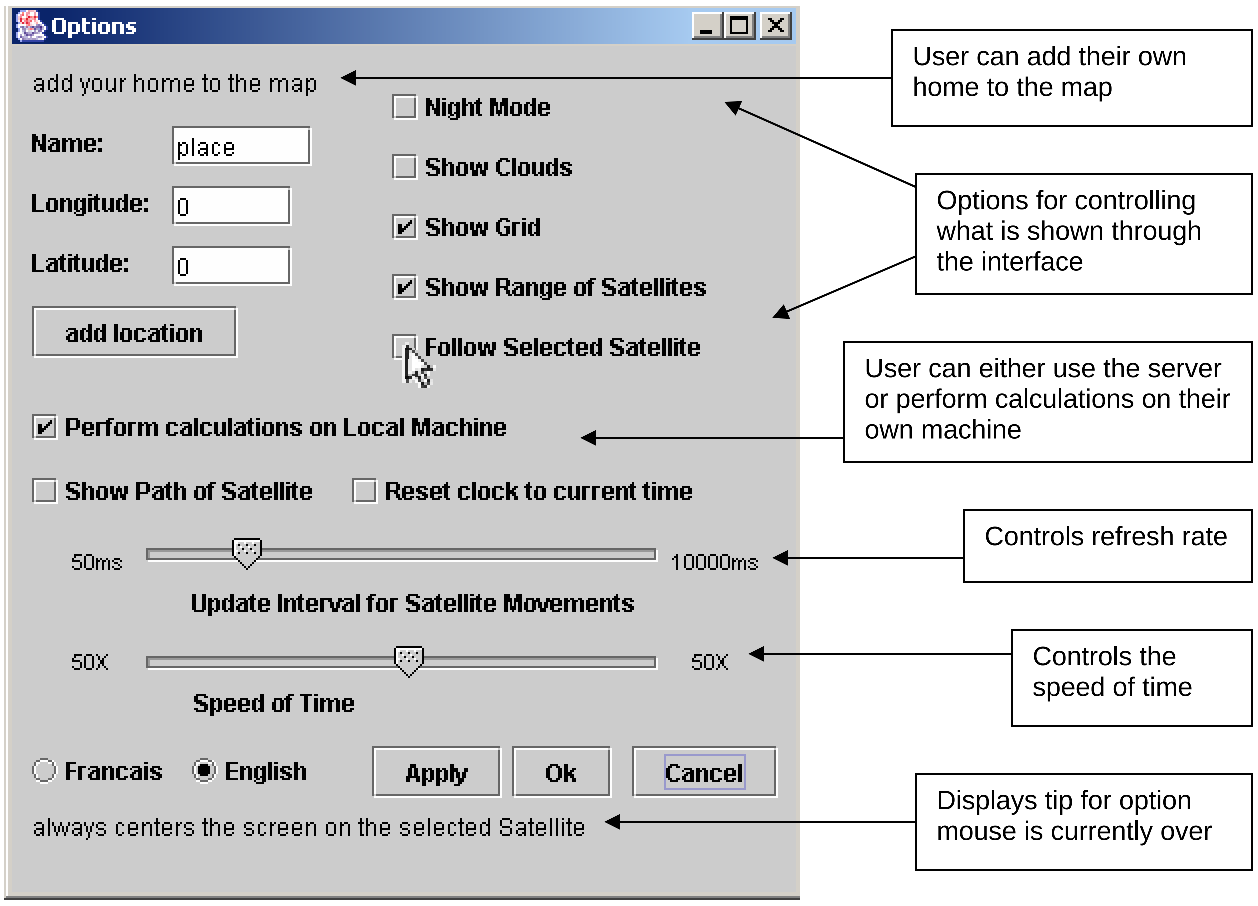


Figure 5. Directory hierarchy of Java source files

6 – Interface Interactions

All interactive features are mouse-controlled and keyboard independent. The controls for basic interaction with the map and selecting a satellite are directly accessible through mouse clicks and drags. Objects such as satellites and paths can be selected using clicks. Double clicks zoom while right clicks zoom out. Mouse drags control the movement of the map while zoomed. These controls are designed to be intuitive and model other GUI-based interfaces.

However, more advanced controls, such as controlling the time, and selecting which interface elements to show are available through an options dialog (Figure 6). The options dialog was developed using Java 2's new Swing architecture. Elements of the interface can be controlled so that the user sees only what is desired. This single options dialog is usable for all the versions of SatMapper, including the low-resolution version and the application version. Unavailable features form special versions are automatically disabled and simple help advice is given on the bottom of the window to help users understand the features.



Options

add your home to the map

Name: place

Longitude: 0

Latitude: 0

add location

[ ] Night Mode

[ ] Show Clouds

[✓] Show Grid

[✓] Show Range of Satellites

[ ] Follow Selected Satellite

[✓] Perform calculations on Local Machine

[ ] Show Path of Satellite

[ ] Reset clock to current time

50ms — 10000ms

Update Interval for Satellite Movements

50X — 50X

Speed of Time

Francais [✓ English]

Apply | Ok | Cancel

always centers the screen on the selected Satellite

User can add their own home to the map

Options for controlling what is shown through the interface

User can either use the server or perform calculations on their own machine

Controls refresh rate

Controls the speed of time

Displays tip for option mouse is currently over

Figure 6. Options Menu

6.1 – Zoom Interaction

Zooming is the process by which SatMapper enlarges or shrinks the size of the Geographic Plane relative to the screen. Therefore, at a higher zoom, the user would see a more precise view of the positions of objects on the grid. However, they would be seeing a smaller part of the Geographic Plane.

SatMapper is able to zoom in on a particular location on the map by double-clicking the mouse. There are 4 levels of zoom: 1, 2, 4 and 6. To increase the zoom-level, the user would double click again on any part of the map. To decrease the zoom-level, the user would right click anywhere on the map.

The whole process is contained within an object called Zoomer (Figure 7). SatMapper sends to Zoomer the Geographic coordinates for of the zoom and a zoom level. Zoomer starts a thread that sends back commands to SatMapper, which control the smooth transition and zooming of the map.

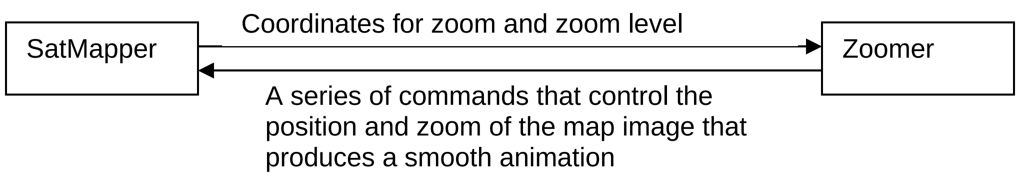

SatMapper
Coordinates for zoom and zoom level
Zoomer
A series of commands that control the position and zoom of the map image that produces a smooth animation

Figure 7. Zoomer Object

The map is separated into 2 layers: the map Image, and the objects (including the grid) on the map. The zooming thread produces 2 animations that run successively; first, the map is zoomed, and then the map is zoomed. The two animations are shown in Figure 8.

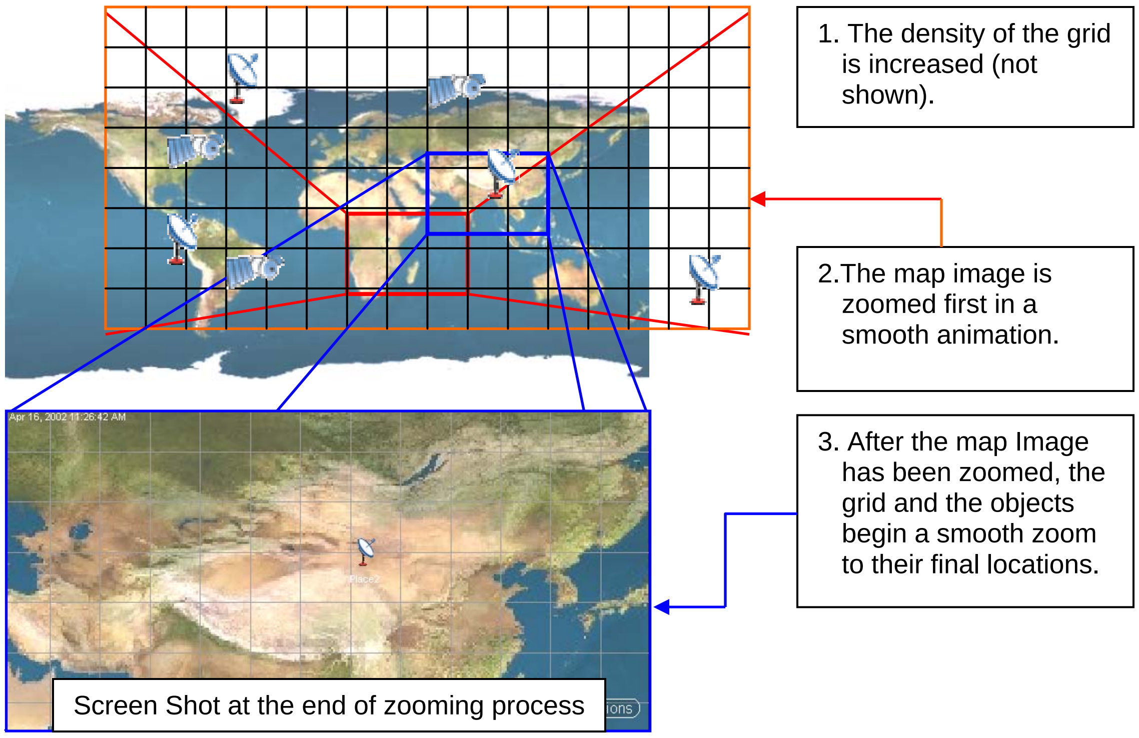

1. The density of the grid is increased (not shown).
2.The map image is zoomed first in a smooth animation.
3. After the map Image has been zoomed, the grid and the objects begin a smooth zoom to their final locations.
Apr 16, 2002 11:26:42 AM
Screen Shot at the end of zooming process

Figure 8. Zooming Thread

The first part of the zooming process is increasing the density of the grid. Because the zoom level is higher, a lesser part of the map is seen and the spaces between the gridlines would increase. The zooming thread increases the density of the grid gradually so that when the process is complete, the same number of grid lines is visible on the screen as when the zoom started.

The second part of the zoom process is to zoom the map image. The size of the map image is increased smoothly and linearly. The increase in size is proportional to the increase in zoom level **(2)**.

Final Size of Map / Initial Size of Map = Final Zoom Level / Initial Zoom Level **(2)**

The position of the screen is moved in synchronization with the enlargement of the map. The final result is a smooth transition from the initial position and zoom to the final position and zoom; there are no sudden movements or jerks in the animation.

The third part of the zoom process is the zooming of the grid and objects. Unlike the map zoom, the images of the objects are not actually enlarged; their positions relative to the screen are stretched to give the effect of zoom. The rest of the process is very much like the map zoom, the size of the Geographic grid is increased and the screen position moved accordingly.

The three steps are purposely performed in sequence one after another. They are done in order one after another simply for visual effect; these steps will work just as well if they happened concurrently.

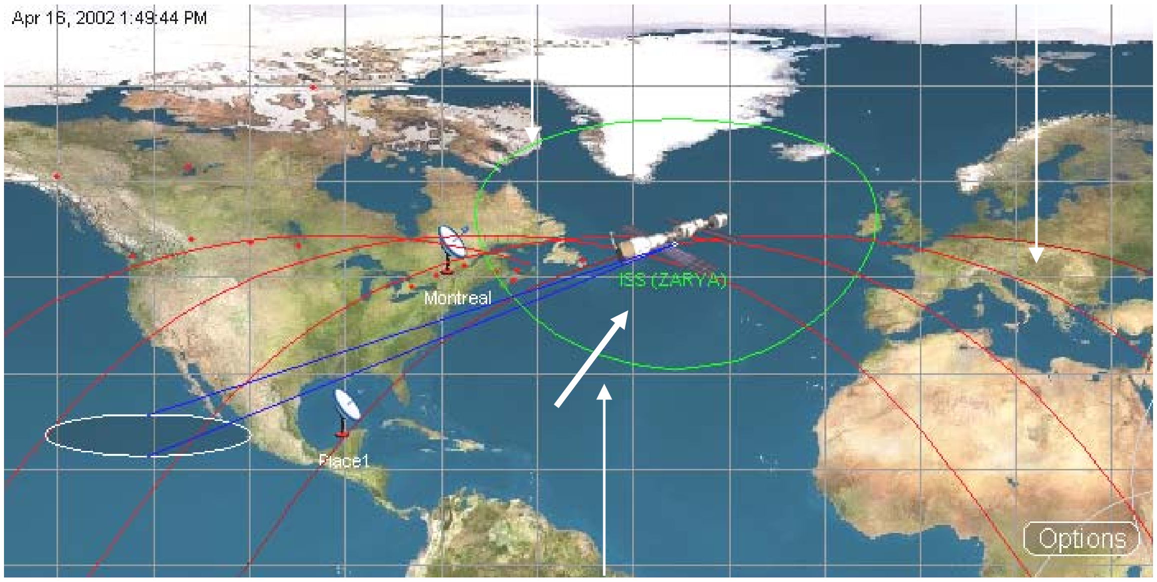
6.2 – Selecting a Satellite

The user is able to select a satellite by clicking on it using a mouse. This indicates the user's desire to focus more attention on the selected satellite. This will cause the interface to take the following steps:

1. The line of sight circle and the name of the selected satellite become green.
2. If orbit paths are enabled in options, it will be shown only for the selected satellite. If no satellite is selected, it is not shown. This prevents an overcrowding of the screen.
3. The interface will call an object called MapSlider. MapSlider starts a thread that produces an animation that slides the screen until the selected satellite is at the center.
4. At the same instance, another thread will produce an animation that results in the creation of the Satellite information display. This display is superimposed on the map and displays orbital parameters such as coordinates, velocity and speed.
5. After MapSlider centers the satellite, SatMapper will follow the satellite and its movements until the user moves the map.

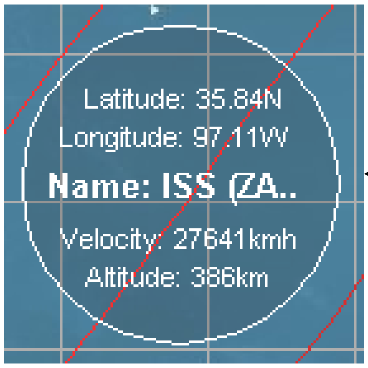
1. The line of sight circle and the name become green

2. The orbit paths are shown



3. The screen moves towards the selected Satellite in an animation

5. The screen will follow the satellite after it is centered

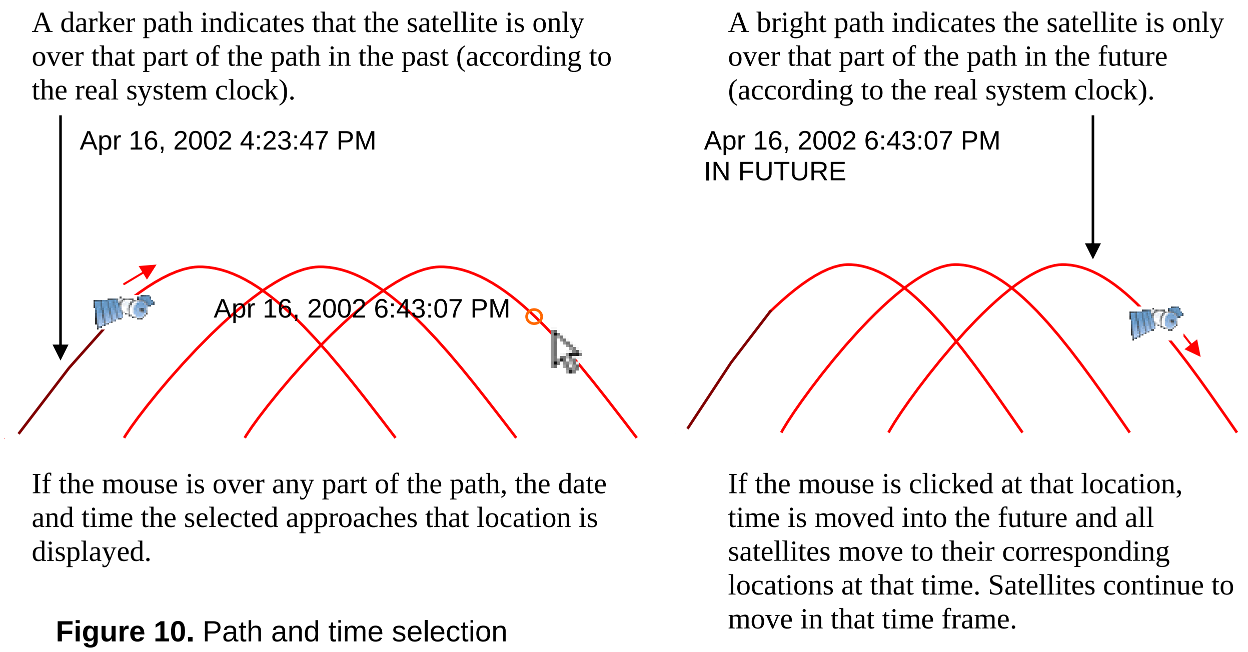

Latitude: 35.84N
Longitude: 97.11W
Name: ISS (ZA..
Velocity: 27641kmh
Altitude: 386km

4. The satellite information display appears in an animation. Orbital parameters are displayed.

Figure 9. Process of selecting a satellite

6.3 – Orbit Path and Time Interaction

Time interaction and calculation of the orbit path require all calculations to be done on the client. This is because the server only sends coordinates for satellites at that specific moment in time. If time interaction is required, SatMapper automatically discontinues use of the server.


A darker path indicates that the satellite is only over that part of the path in the past (according to the real system clock).

A bright path indicates the satellite is only over that part of the path in the future (according to the real system clock).

If the mouse is over any part of the path, the date and time the selected approaches that location is displayed.

If the mouse is clicked at that location, time is moved into the future and all satellites move to their corresponding locations at that time. Satellites continue to move in that time frame.

Figure 10. Path and time selection

The path is stored in a 3 dimensional array. For every coordinate, it contains a longitude, latitude, and the absolute time. SatMapper uses this array to determine the time frame for every point in the path. If the user moves the mouse over any part of the path it will display the time that at which the satellite will reach that coordinate.

The path contains coordinates for the next 3 hours of the satellite's trek. It would be inefficient for it to be generated constantly; so the path coordinates are recalculated only when time changes by more than 30 minutes.

In addition to selecting time, it is possible to control the speed of time through the creation of a timer whose speed is controlled by SatMapper. This timer is an object called SatTimer. The system clock is simply used as a base for the starting time, and SatTimer takes over the normal clock function. If the user changes time, the base time and SatTimer have to be reinitialized in order to continue to give accurate data.

This time interaction is important because it allows the user to predict when the next pass of a satellite may be. Also, controlling the speed of time allows users the watch the movements of many satellites at once. These features are disabled on default and are available in SatMapper's options menu (Figure 6).

6.4 – Switching Map Images

Another feature quite unique to SatMapper is the ability to switch maps. Included in SatMapper are special interest maps such as one of the world at night and with clouds.

All the map images are obtained from NASA`s Visible Earth Web Site **[2]**. The program allows the use of its images for non-commercial public uses. The copyright policy can be found in *Appendix D*. These images were chosen because they were composed from actual photos of the earth and gave a natural look. The implementation of additional maps, such as political maps, could easily be done.

The architecture of SatMapper allows the seamless switching of these maps through transparencies. As in the figure below, you can see the world in different stages of day and night. In the same way, with transparencies, SatMapper can give the visual effect of traveling through clouds when zooming. At this stage, these special maps are enabled only when the user requests it through the options menu. Figure 11 offers some screenshots of SatMapper with different maps and at different stages of transition.

It is possible in the future to have the maps represent the real world. For example, the clouds image could be updated daily, from weather satellites.

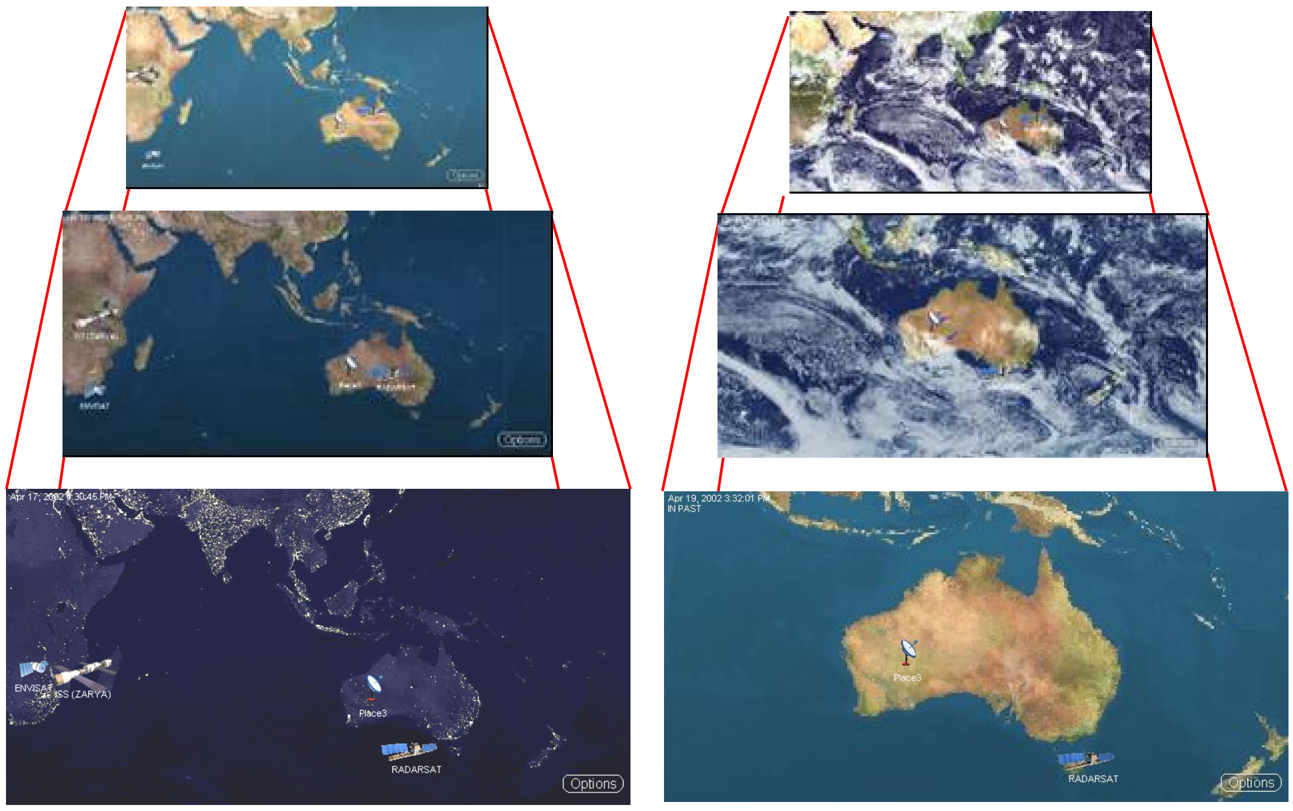


Figure 11. Map transitions using transparency

7 – Coordinate System Used by Interface

The movement of the screen around the map is made possible by the existence of two coordinate systems. All objects: places and satellites store their coordinates on the Geographic Plane in terms of longitude and latitude. Some non-dynamic interface elements such as informational text do not move, so they are represented by screen coordinates (distance in pixels from the edge of the screen). The two coordinates systems have been implemented in a way that they can be dynamically transferred from one to another. Whenever objects are drawn, their Geographic coordinates must be converted first to screen coordinates, then drawn on the screen.

Because of the many animations and interactive nature of the interface, the actual representation of the Geographic coordinates in the interface is dynamic. This means that at the screen coordinates of a Geographic coordinate could be changing, as the screen is zooming or moving.

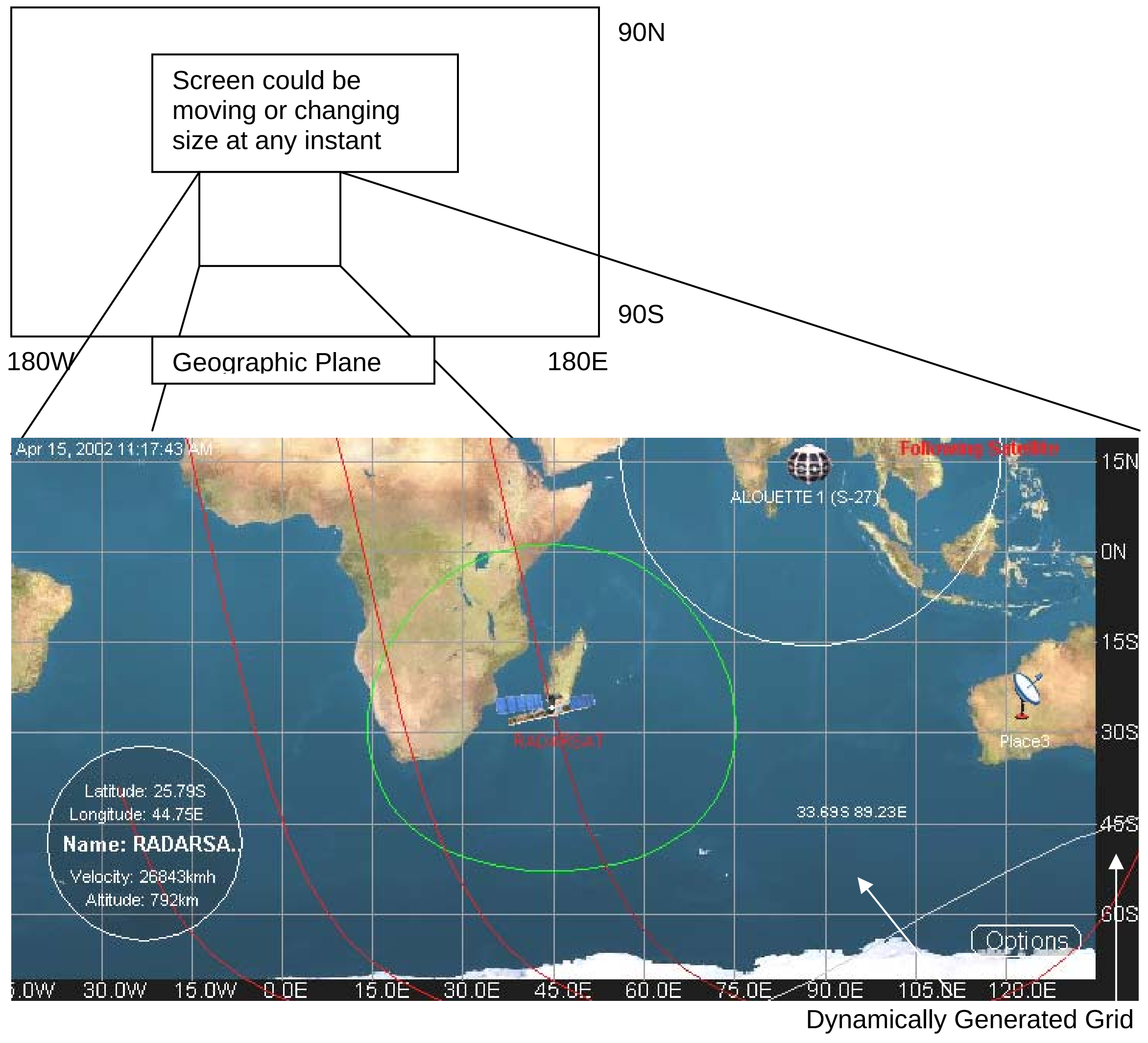


Figure 12. Movement of interface in Geographic Plane

All objects on the screen such as the grid and Satellites have to have their screen coordinates recalculated every time the screen is repainted. Their positions are dynamically calculated through several functions that convert between the screen and Geographic coordinates. They are encapsulated in two functions and getXCoord() and getYCoord.

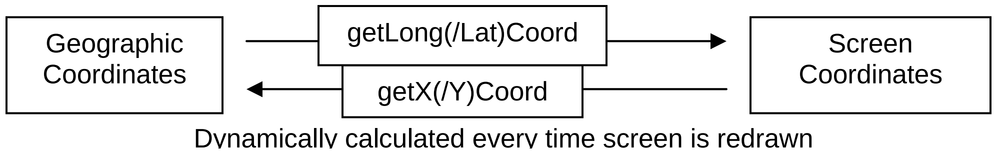


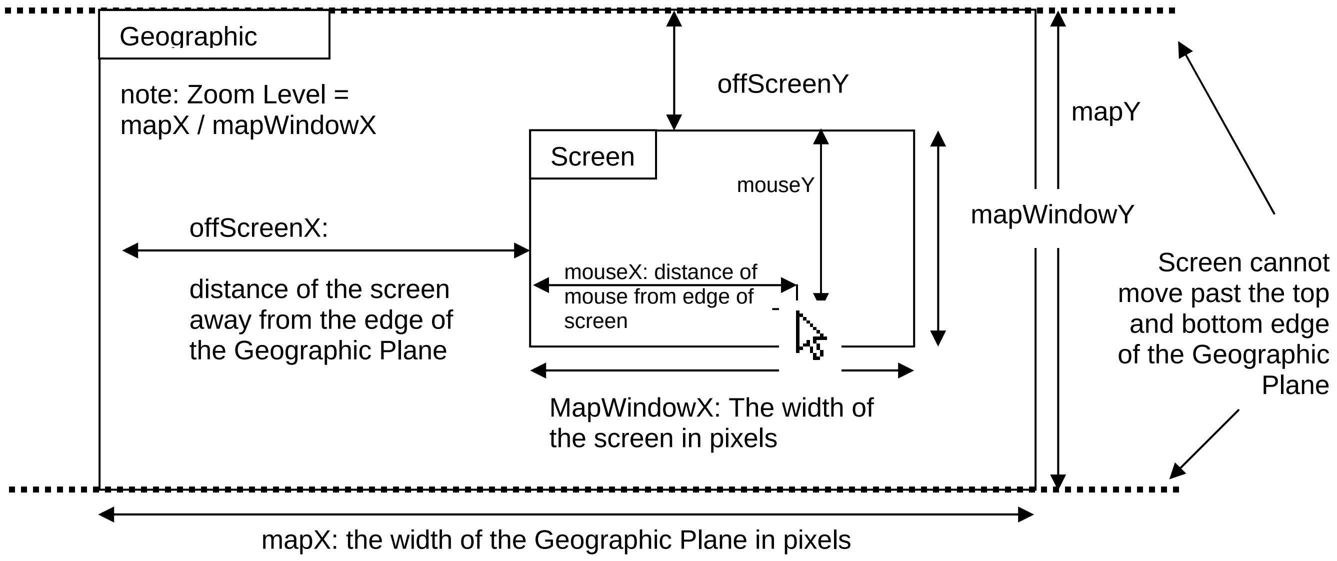
Figure 13. Converting between two Coordinate Systems

Correspondingly, because the interface needs to interact with the objects, there is also a system for converting screen coordinates to Geographic coordinates. These calculations are encapsulated in methods in SatMapper called getLongCoord() and getLatCoord().

For example, the mouse coordinate is given by the Java system in terms of pixel distance from the edge of the screen. To get the longitude of that coordinate, send the x coordinate to getLongCoord and it would return the longitudinal coordinate.

7.1 Representing Screen Position

Because of the dynamic nature of the Interface, and for efficiency purposes, the position of the screen in relation to the map could not be effectively represented by Geographic coordinates. A system was developed for SatMapper that involved positioning the screen by changing a value called offScreenX and offScreenY. These two values represent the distance the screen is away from the left and upper edge of the map. All SatMapper dynamic interface calculations use these values in a simple calculation to convert Geographic coordinates to screen coordinates.



Geographic
note: Zoom Level = mapX / mapWindowX
offScreenY
mapY
Screen
mouseY
mapWindowY
offScreenX:
distance of the screen away from the edge of the Geographic Plane
mouseX: distance of mouse from edge of screen
Screen cannot move past the top and bottom edge of the Geographic Plane
MapWindowX: The width of the screen in pixels
mapX: the width of the Geographic Plane in pixels

Figure 14. Different ways screen coordinates are measured

7.2 – Properties of the Geographic Plane

There are many intricacies involved in dealing with the Geographic Plane because it is a system that represents a spherical Earth in 2d. The latitudinal lines on the Geographic Plane are continuous. SatMapper's interface represents this anomaly by creating a map that can be rotated continuously along the horizontally axis. This effect is illustrated in Figure 15.

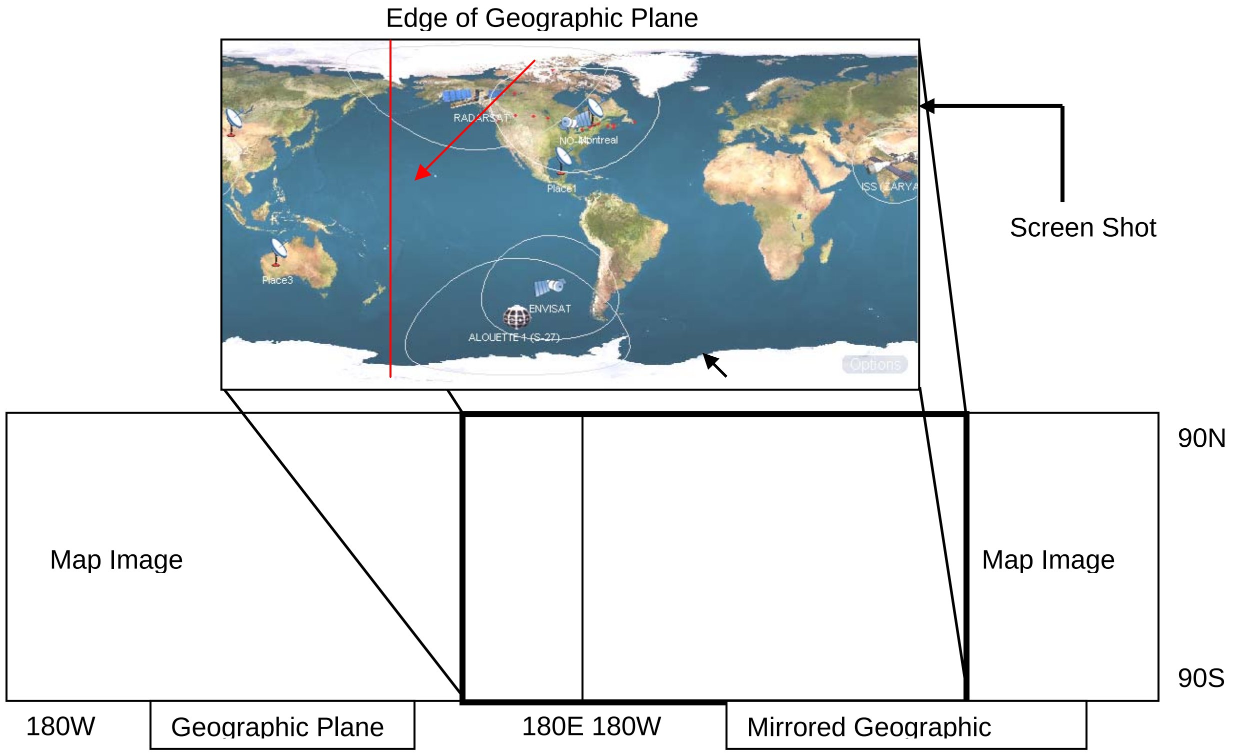


Figure 15. The continuous nature of the Geographic Plane

7.3 – Orbits Paths and Line of Sight in Geographic Plane

Another property of the Geographic grid is that none of the longitudinal lines are parallel. In a sphere, the longitudinal lines converge at the poles. However, on a Geographic Plane, longitudinal lines are shown as parallel (Figure 16). Therefore lines near the equator would appear different near the poles.

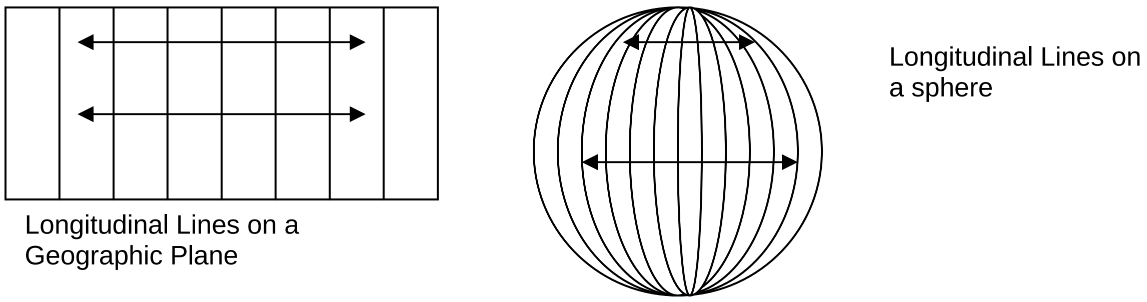

Longitudinal Lines on a sphere
Longitudinal Lines on a Geographic Plane

Figure 16. Longitudinal lines on the Geographic Plane and on a sphere

This property will cause the representation of satellite orbits paths and line of sight circles slightly more difficult. Figure 17 shows the actual and screen representations of Orbit Path and Line of Sight.

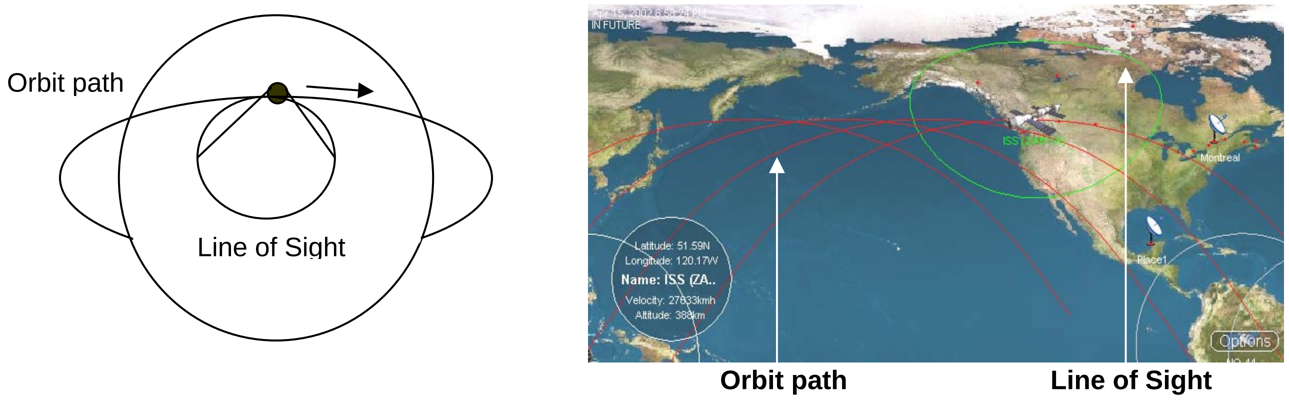


Figure 17. Actual and Screen representations of Orbit Path and Line of Sight

The line of sight of a satellite is the area on the ground where the satellite is able to form a straight communications link with a satellite. Assuming that the ground is flat, the line of sight should form a conical cross-section on the earth. Every point on the outer rim of the line of sight should be equidistant from the center point, or the Geographic coordinates of the satellite.

Using a 3d-geometrical formula **[8]** that calculates new coordinates based on a starting point, direction (angle of travel), and distance. By repeating this calculation for all the angles in a circle, the line of sight area could be represented by a continuous circular outline. This functionality is incorporated into a range object that is associated with every satellite. And it is updated every time the satellite moves.

Connecting a series satellite coordinates for a certain time period creates the path. The subject of calculating satellite coordinates is covered in section 5 of this report. These coordinates are updated at a set time interval, or every 30minutes.

8 – Web Site Development

A website was developed in preparation for the deployment of SatMapper on the web. Since the aim of SatMapper is for use by the general public, accessibility is a key concern. Therefore, three different version of SatMapper were created for different computers. These three versions were linked to a home page, which is shown in Figure 18.

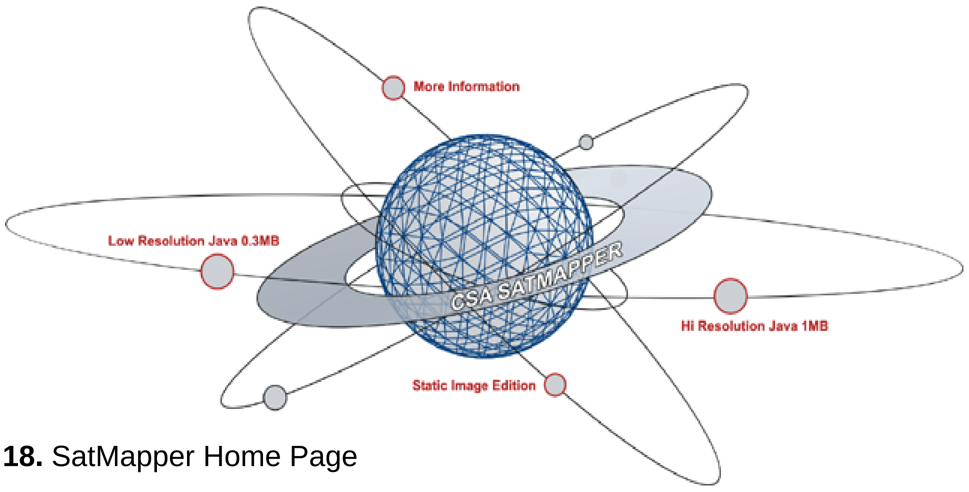


Figure 18. SatMapper Home Page

People with low-speed modem access can use the low-resolution of version of SatMapper, which contains only one map image at a lower resolution. For people who cannot run java at all, a static-image edition was created. This version uses the jpeg encoder to write a static image of SatMapper. This version is not interactive but is the most accessible and smallest in size of the three versions. With these three pages, SatMapper will be easily accessible to all Internet users. All three versions of SatMapper is enclosed in a frame that contains a title and short instructions; it is shown in Figure 19.

In the Java-based pages, there is JavaScript code that allows the browser automatically download the Java 2 Plug-in with the user's permission. This code was generated using Sun Microsystems's HTML converter.

All the graphics in this website were created using Corel CorelDraw and Adobe Photoshop. The HTML was generated using Macromedia Dreamweaver. The sphere on the home page was generated using 3DS-Max.

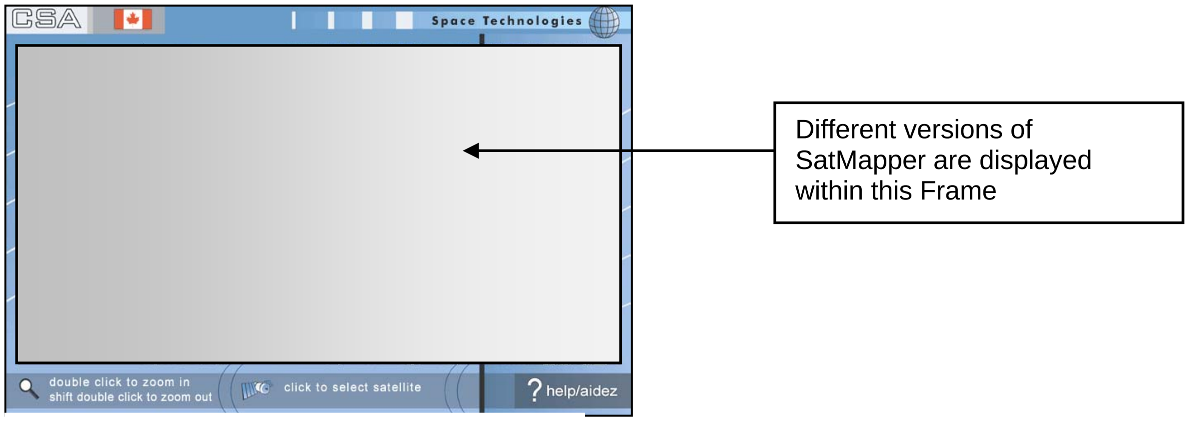

CSA
Space Technologies
double click to zoom in
shift double click to zoom out
click to select satellite
? help/aidez
Different versions of SatMapper are displayed within this Frame

Figure 19. SatMapper Web Page

9 – Testing and Verification

It was very important for SatMapper have accurate orbital prediction. To test it, SatMapper was run in parallel with other established Satellite-tracking applications such as NASA's J-Track NLSA's Nova. It was determined that SatMapper accurately presented satellite coordinates.

SatMapper has been tested with actual passes of the International Space Station. When the ISS came within range on SatMapper, the JLoad Groundstation software reported signal acquisition and was able to initiate a communications link. When the ISS left the range of the groundstation on SatMapper, the groundstation reported signal loss. This proved SatMapper was able to accurately present the pass of the ISS.

The Internet version of SatMapper was tested on a test server. A server was made which hosted SatMapper and its web page. The server run on the Intranet and allowed the testing of SatMapper with a wide range of computers.

SatMapper was presented to members of the QuickSat team, the director of Space Technologies and then to IT. It was received well and some suggestions were taken into account; no problems were reported.

There are no known problems with SatMapper and it has been shown to be accurate and reliable.

References

[1] NASA, *J-Track Satellite Tracking*,
URL: http://liftoff.msfc.nasa.gov/RealTime/JTrack/ (current April 24, 2002).

[2] NASA, *Visible Earth*,
URL: http://visibleearth.nasa.gov/ (April 24, 2001).

[3] Weeks, James. *JPEG Encoder*,
URL: http://www.obrador.com/essentialjpeg/jpeg.htm (current April 24, 2002).

[4] NORAD, *NORAD Two-Line Element Sets Current Data*,
http://www.celestrak.com/NORAD/elements/ (current April 24, 2002).

[5] Hoots, Felix, and Roehrich, Ronald, *SpaceTrack Report No. 3: Models for Propagation of NORAD Element Sets*, Department of Defense, 1988.

[6] The Radio Amateur Satellite Corporation. *Keplerian Data Formats,*
URL: http://www.amsat.org/amsat/keps/formats.html (current April 24, 2002).

[7] The Math Forum at Drexel, *Longitude and Latitude to Determine Distance*,
URL: http://mathforum.org/dr.math/problems/atlas5.21.96.html (current April 24, 2002).

[8] The Math Forum at Drexel, *Walking Around the World*,
URL: http://mathforum.org/dr.math/problems/kluj2.8.97.html (current April 24, 2002).

APPENDIX A

Copyright notice for map images from NASA Visible Earth

NASA, *Visible Earth Help*, < URL: http://visibleearth.nasa.gov/help.html > (April 24, 2002).

APPENDIX B

Appendix B is Obtained From:
Hoots, Felix and Roehrich, Ronald. *SpaceTrack Report No. 3: Models for Propagation of NORAD Element Sets*, Department of Defense, 1988.

These are the formulas used to calculate the orbital parameters of satellites.

6 THE SGP4 MODEL

The NORAD mean element sets can be used for prediction with SGP4. All symbols not defined below are defined in the list of symbols in Section Twelve. The original mean motion (n''_o) and semimajor axis (a''_o) are first recovered from the input elements by the equations

$$a_1 = \left(\frac{k_e}{n_o}\right)^{\frac{2}{3}}$$

$$\delta_1 = \frac{3}{2}\frac{k_2}{{a_1}^2}\frac{(3\cos^2 i_o - 1)}{(1-{e_o}^2)^{\frac{3}{2}}}$$

$$a_o = a_1\left(1 - \frac{1}{3}\delta_1 - {\delta_1}^2 - \frac{134}{81}{\delta_1}^3\right)$$

$$\delta_o = \frac{3}{2}\frac{k_2}{{a_o}^2}\frac{(3\cos^2 i_o - 1)}{(1-{e_o}^2)^{\frac{3}{2}}}$$

$$n''_o = \frac{n_o}{1+\delta_o}$$

$$a''_o = \frac{a_o}{1-\delta_o}.$$

For perigee between 98 kilometers and 156 kilometers, the value of the constant s used in SGP4 is changed to

$$s^* = a''_o(1-e_o) - s + a_E$$

For perigee below 98 kilometers, the value of s is changed to

$$s^* = 20/\text{XKMPER} + a_E.$$

If the value of s is changed, then the value of $(q_o - s)^4$ must be replaced by

$$(q_o - s^*)^4 = \left[[(q_o-s)^4]^{\frac{1}{4}} + s - s^*\right]^4.$$

Then calculate the constants (using the appropriate values of s and $(q_o - s)^4$)

$$\theta = \cos i_o$$

$$\xi = \frac{1}{a_o'' - s}$$

$$\beta_o = (1 - e_o{}^2)^{\frac{1}{2}}$$

$$\eta = a_o'' e_o \xi$$

$$C_2 = \ (q_o - s)^4 \xi^4 n_o'' (1-\eta^2)^{-\frac{7}{2}} \left[a_o'' \left(1 + \frac{3}{2}\eta^2 + 4e_o\eta + e_o\eta^3 \right) + \frac{3}{2} \frac{k_2 \xi}{(1-\eta^2)} \left(-\frac{1}{2} + \frac{3}{2}\theta^2 \right) (8 + 24\eta^2 + 3\eta^4) \right]$$

$$C_1 = B^* C_2$$

$$C_3 = \frac{(q_o - s)^4 \xi^5 A_{3,0} n_o'' a_E \sin i_o}{k_2 e_o}$$

$$C_4 = \ 2n_o''(q_o - s)^4 \xi^4 a_o'' \beta_o{}^2 (1-\eta^2)^{-\frac{7}{2}} \left(\left[2\eta(1 + e_o\eta) + \frac{1}{2}e_o + \frac{1}{2}\eta^3 \right] - \frac{2k_2\xi}{a_o''(1-\eta^2)} \times \left[3(1-3\theta^2) \left(1 + \frac{3}{2}\eta^2 - 2e_o\eta - \frac{1}{2}e_o\eta^3 \right) + \frac{3}{4}(1-\theta^2)(2\eta^2 - e_o\eta - e_o\eta^3)\cos 2\omega_o \right] \right)$$

$$C_5 = 2(q_o - s)^4 \xi^4 a_o'' \beta_o{}^2 (1-\eta^2)^{-\frac{7}{2}} \left[1 + \frac{11}{4}\eta(\eta + e_o) + e_o\eta^3 \right]$$

$$D_2 = 4a_o'' \xi C_1{}^2$$

$$D_3 = \frac{4}{3} a_o'' \xi^2 (17a_o'' + s) C_1{}^3$$

$$D_4 = \frac{2}{3} a_o'' \xi^3 (221a_o'' + 31s) C_1{}^4.$$

The secular effects of atmospheric drag and gravitation are included through the equations

$$M_{DF} = M_o + \left[1 + \frac{3k_2(-1+3\theta^2)}{2a_o''^2 \beta_o{}^3} + \frac{3k_2{}^2(13 - 78\theta^2 + 137\theta^4)}{16a_o''^4 \beta_o{}^7} \right] n_o''(t - t_o)$$

$$\omega_{DF} = \ \omega_o + \left[-\frac{3k_2(1-5\theta^2)}{2a_o''^2 \beta_o{}^4} + \frac{3k_2{}^2(7 - 114\theta^2 + 395\theta^4)}{16a_o''^4 \beta_o{}^8} + \frac{5k_4(3 - 36\theta^2 + 49\theta^4)}{4a_o''^4 \beta_o{}^8} \right] n_o''(t - t_o)$$

APPENDIX B

$$\Omega_{DF} = \Omega_o + \left[-\frac{3k_2\theta}{a_o''^2{\beta_o}^4} + \frac{3{k_2}^2(4\theta - 19\theta^3)}{2a_o''^4{\beta_o}^8} + \frac{5k_4\theta(3-7\theta^2)}{2a_o''^4{\beta_o}^8} \right] n_o''(t-t_o)$$

$$\delta\omega = B^* C_3 (\cos\omega_o)(t-t_o)$$

$$\delta M = -\frac{2}{3}(q_o - s)^4 B^* \xi^4 \frac{a_E}{e_o\eta}[(1+\eta\cos M_{DF})^3 - (1+\eta\cos M_o)^3]$$

$$M_p = M_{DF} + \delta\omega + \delta M$$

$$\omega = \omega_{DF} - \delta\omega - \delta M$$

$$\Omega = \Omega_{DF} - \frac{21}{2}\frac{n_o'' k_2 \theta}{a_o''^2{\beta_o}^2} C_1 (t-t_o)^2$$

$$e = e_o - B^* C_4 (t-t_o) - B^* C_5 (\sin M_p - \sin M_o)$$

$$a = a_o''[1 - C_1(t-t_o) - D_2(t-t_o)^2 - D_3(t-t_o)^3 - D_4(t-t_o)^4]^2$$

$$\begin{aligned} I\!L = \; & M_p + \omega + \Omega + n_o'' \left[\frac{3}{2} C_1 (t-t_o)^2 + (D_2 + 2{C_1}^2)(t-t_o)^3 \right. \\ & + \frac{1}{4}(3D_3 + 12C_1D_2 + 10{C_1}^3)(t-t_o)^4 \\ & \left. + \frac{1}{5}(3D_4 + 12C_1D_3 + 6{D_2}^2 + 30{C_1}^2D_2 + 15{C_1}^4)(t-t_o)^5 \right] \end{aligned}$$

$$\beta = \sqrt{(1-e^2)}$$

$$n = k_e \Big/ a^{\frac{3}{2}}$$

where $(t - t_o)$ is time since epoch. It should be noted that when epoch perigee height is less than 220 kilometers, the equations for a and $I\!L$ are truncated after the C_1 term, and the terms involving C_5, $\delta\omega$, and δM are dropped.

Add the long-period periodic terms

$$a_{xN} = e\cos\omega$$

APPENDIX B

$$I\!L_L = \frac{A_{3,0}\sin i_o}{8k_2 a\beta^2}(e\cos\omega)\left(\frac{3+5\theta}{1+\theta}\right)$$

$$a_{yNL} = \frac{A_{3,0}\sin i_o}{4k_2 a\beta^2}$$

$$I\!L_T = I\!L + I\!L_L$$

$$a_{yN} = e\sin\omega + a_{yNL}.$$

Solve Kepler's equation for $(E+\omega)$ by defining

$$U = I\!L_T - \Omega$$

and using the iteration equation

$$(E+\omega)_{i+1} = (E+\omega)_i + \Delta(E+\omega)_i$$

with

$$\Delta(E+\omega)_i = \frac{U - a_{yN}\cos(E+\omega)_i + a_{xN}\sin(E+\omega)_i - (E+\omega)_i}{-a_{yN}\sin(E+\omega)_i - a_{xN}\cos(E+\omega)_i + 1}$$

and

$$(E+\omega)_1 = U.$$

The following equations are used to calculate preliminary quantities needed for short-period periodics.

$$e\cos E = a_{xN}\cos(E+\omega) + a_{yN}\sin(E+\omega)$$

$$e\sin E = a_{xN}\sin(E+\omega) - a_{yN}\cos(E+\omega)$$

$$e_L = ({a_{xN}}^2 + {a_{yN}}^2)^{\frac{1}{2}}$$

$$p_L = a(1-{e_L}^2)$$

$$r = a(1-e\cos E)$$

APPENDIX B

$$\dot{r} = k_e \frac{\sqrt{a}}{r} e \sin E$$

$$r\dot{f} = k_e \frac{\sqrt{p_L}}{r}$$

$$\cos u = \frac{a}{r}\left[\cos(E+\omega) - a_{xN} + \frac{a_{yN}(e\sin E)}{1+\sqrt{1-{e_L}^2}}\right]$$

$$\sin u = \frac{a}{r}\left[\sin(E+\omega) - a_{yN} - \frac{a_{xN}(e\sin E)}{1+\sqrt{1-{e_L}^2}}\right]$$

$$u = \tan^{-1}\left(\frac{\sin u}{\cos u}\right)$$

$$\Delta r = \frac{k_2}{2p_L}(1-\theta^2)\cos 2u$$

$$\Delta u = -\frac{k_2}{4{p_L}^2}(7\theta^2-1)\sin 2u$$

$$\Delta\Omega = \frac{3k_2\theta}{2{p_L}^2}\sin 2u$$

$$\Delta i = \frac{3k_2\theta}{2{p_L}^2}\sin i_o \cos 2u$$

$$\Delta\dot{r} = -\frac{k_2 n}{p_L}(1-\theta^2)\sin 2u$$

$$\Delta r\dot{f} = \frac{k_2 n}{p_L}\left[(1-\theta^2)\cos 2u - \frac{3}{2}(1-3\theta^2)\right]$$

The short-period periodics are added to give the osculating quantities

$$r_k = r\left[1 - \frac{3}{2}k_2\frac{\sqrt{1-{e_L}^2}}{{p_L}^2}(3\theta^2-1)\right] + \Delta r$$

$$u_k = u + \Delta u$$

$$\Omega_k = \Omega + \Delta\Omega$$

$$i_k = i_o + \Delta i$$

$$\dot{r}_k = \dot{r} + \Delta\dot{r}$$

$$r\dot{f}_k = r\dot{f} + \Delta r\dot{f}.$$

Then unit orientation vectors are calculated by

$$\mathbf{U} = \mathbf{M}\sin u_k + \mathbf{N}\cos u_k$$

$$\mathbf{V} = \mathbf{M}\cos u_k - \mathbf{N}\sin u_k$$

where

$$\mathbf{M} = \left\{ \begin{array}{l} M_x = -\sin\Omega_k \cos i_k \\ M_y = \cos\Omega_k \cos i_k \\ M_z = \sin i_k \end{array} \right\}$$

$$\mathbf{N} = \left\{ \begin{array}{l} N_x = \cos\Omega_k \\ N_y = \sin\Omega_k \\ N_z = 0 \end{array} \right\}.$$

Then position and velocity are given by

$$\mathbf{r} = r_k\mathbf{U}$$

and

$$\dot{\mathbf{r}} = \dot{r}_k\mathbf{U} + (r\dot{f})_k\mathbf{V}.$$

A FORTRAN IV computer code listing of the subroutine SGP4 is given below. These equations contain all currently anticipated changes to the SCC operational program. These changes are scheduled for implementation in March, 1981.