OSGeo Journal Volume 8

February 2011

FOSS4G 2009
Conference Proceedings

OSGeo Community
News & Announcements
Case Studies
Integration Examples

FOSS4G
DENVER 2011
SEPTEMBER 12-16

The Annual International

FREE & OPEN SOURCE
SOFTWARE FOR GEOSPATIAL

Conference Event

2011.FOSS4G.ORG

OSGeo
Your Open Source Compass

Volume 8 Contents

From the Editor

OSGeo has just past its 5th birthday, along with this 8th volume of the OSGeo Journal! With this edition we bring a few news headlines from the past couple months, a few general articles and, most significantly, several top papers from the **FOSS4G 2009** conference event held in Sydney, Australia.

The Journal has become a diverse platform for several groups and growth in each area is expected to continue. The key groups that read and contribute to the Journal include software developers sharing information about their projects or communities, power users showing off their solutions, academia seeking to publish their research and observations in a peer-reviewed, open source friendly medium. OSGeo also uses the Journal to share community updates and the annual reports of the organisation.

Welcome to those of you who are new to the OSGeo Journal. Our Journal team and volunteer reviewers and editors hope you enjoy this volume. We also invite you to submit your own articles to any of our various sections. To submit an article, register as an "author" and sign in at `http://osgeo.org/ojs`. Then when you log in you will see an option to submit an article.[1]

We look forward to working with, and for, you in the upcoming year. It's sure to be an interesting year as we see OSGeo, Open Source in general and all our relate communities continue to grow. Nowhere else is this growth more apparent than at our annual conference: **FOSS4G 2011 Denver**, September, 2011.[2] Keep an eye on your OSGeo mailing lists, blogs and other feeds to follow the latest FOSS4G announcements, including the invitation to submit presentation proposals.[3] It will be as competitive as ever to get a speaking slot, so be sure to make your title and abstract really stand out.

Wishing you the best for 2011 and hoping to see you in Denver!

[illegible]

Tyler Mitchell
`tmitchell@osgeo.org`
Editor in chief, OSGeo Journal
Executive Director, OSGeo

[1]The direct URL for article submission is: `https://www.osgeo.org/ojs/index.php/journal/author/submit`
[2]FOSS4G 2011 Denver: `http://2011.foss4g.org`
[3]FOSS4G 2011 Abstract Submission: `http://2011.foss4g.org/program`

FOSS4G 2009 Conference Proceedings

From the Academic Track Chair

Prof. Thierry Badard

The FOSS4G 2009 academic track aimed to bring together researchers, developers, users and practitioners – all who were carrying out research and development in the free and open source geospatial fields and who were willing to share original, recent developments and experiences.

The primary goal was to promote cooperative research between OSGeo developers and academia, but the academic track has also acted as an inventory of current research topics. This track was the right forum to highlight the most important research challenges and trends in the domain and let them become the basis for an informal OSGeo research agenda. It has fostered interdisciplinary discussions in all aspects of the free and open source geospatial domains. It was organized to promote networking between the participants, to initiate and favour discussions regarding cutting-edge technologies in the field, to exchange research ideas and to promote international collaboration.

In addition to the OSGeo Foundation[23], the ICA (International Cartographic Association) working group on open source geospatial technologies[24]) was proud to support the organisation of the track.

The coordinators sought to gather paper submissions globally that addressed theoretical, technical, and practical topics related to the free and open source geospatial domain. Suggested topics included, but were not limited to, the following:

- State of the art developments in Open Source GIS
- Open Source GIS in Education
- Interoperability and standards - OGC, ISO/TC 211, Metadata
- Spatial Data Infrastructures and Service Oriented Architectures
- Free and open source Web Mapping, Web GIS and Web processing services
- Cartography and advanced styling
- Earth Observation and remote sensing
- Spatial and Spatio-temporal data, analysis and integration
- Free and Open Source GIS application use cases in Government, Participatory GIS, Location based services, Health, Energy, Water, Urban and Environmental Planning, Climate change, etc.

In response to the call for papers, 25 articles were submitted to the academic track. The submissions were highly diversified, and came from USA, Canada, Thailand, Japan, South Korea, Sri Lanka, Australia, New Zealand, Italy, Denmark, France, Germany, Switzerland, Romania and Turkey. Selection of submissions were based on the full papers received. All submissions were thoroughly peer reviewed by two to three members of the international scientific committee and refereed for their quality, originality and relevance. The scientific committee selected 12 papers (48% acceptance rate) for presentation at the FOSS4G 2009 conference. From those, 6 papers were accepted for presentation in the proceedings of the academic track, which are published in this volume of the OSGeo Journal. They correspond to the 6 best papers assessed by the international scientific committee.

The accepted and published papers covered a wide

[23]OSGeo: Open Source Geospatial Foundation: http://osgeo.org
[24]ICA open source working group: http://ica-opensource.scg.ulaval.ca/

range of cutting-edge research topics and novel applications on Free and Open Source Geospatial technologies. I am particularly proud and happy to see some very high quality scientific contributions published in the OSGeo Journal. This will undoubtedly encourage more interesting research to be published in this volume, as our OSGeo journal is an open access journal. In addition, it helps draw attention to this important project of the OSGeo Foundation. I hope the publication of these proceedings in the OSGeo journal will encourage future scientists, researchers and members of academia to consider the OSGeo Journal as an increasingly valuable place to publish their research works and case studies.

As a concluding note, I would like to take the opportunity to thank the individuals and institutions that made the FOSS4G 2009 academic track possible. First, I would like to thank the international scientific committee members and external reviewers for evaluating the assigned papers in a timely and professional manner. Next, I would like to recognize the tremendous efforts put forward by members of the local organising committee of FOSS4G 2009 for accommodating and supporting the academic track. Finally, I want to thank the authors for their contributions, efforts, patience and support that made this academic track a huge success.

January, 2011
Prof. Thierry Badard
Laval University, Canada
Chair, FOSS4G 2009 Academic Track
Co-chair, ICA Working Group on Open Source Geospatial Technologies

Collaborative Web-Based Mapping of Real-Time Flight Simulator and Sensor Data

Rabih Dagher, Cristian Gadea, Bogdan Ionescu, Dan Ionescu and Robin Tropper

Abstract

Google Maps is an example of how Web 2.0 technology such as AJAX can be used to create online map services that are easy to access, user-friendly and fast. Thanks to flexible web-based mapping APIs, it is now possible for non-experts to plot and distribute GIS (Geographic Information System) data to a large audience. Most data plotted so far, however, has been relatively static. In addition, the typical webpage layout has limited the interaction possibilities for online maps when compared with windowed desktop applications. This paper will present a JEE-based publish/subscribe architecture that allows real-time sensor data to be displayed collaboratively on the web, requiring users to have nothing more than a web browser and Internet connectivity to gain access to that data. The architecture is tested using live data from Microsoft Flight Simulator and data conforming to the OGC Sensor Observation Service (SOS) standard. By using the latest web-based technology from open source projects like OpenLayers and 52North, this paper shows how maps and GIS data can be made more accessible, more social and generally more useful.

Introduction

With the growing adoption of social websites like Twitter, the demand for real-time data on the Internet is now higher than ever before. New web browser technologies have made it easy for users to access and publish dynamic data, such as what they are doing and where they are currently located. However, while there exist online services that promise "real-time" geographic data, a closer look will almost always reveal that the data is delayed or otherwise not presented in a useful way. In addition, collaborating with others in real-time on maps themselves containing real-time data has yet to be attempted from inside a web browser.

This paper introduces a new framework that allows displaying real-time sensor data within a collaborative web-based environment. The framework extends the publish/subscribe messaging model to meet the reliability and scalability demands of a social and collaborative online environment based on real-time data. While the framework can be adapted to a large variety of real-time streaming data sources, this paper will focus on live data provided by Microsoft Flight Simulator 2004 and data that conforms to the Sensor Observation Service (SOS) standard. SOS is one of the Sensor Web Enablement (SWE) standards of the Open Geospatial Consortium (OGC), an international standards organization with nearly 400 supporting companies and institutions (14).

The real-time data will be rendered by using the open source OpenLayers API and will be displayed inside an in-house platform for real-time web-based collaboration known as UC-IC ("you see I see"). UC-IC recreates a familiar desktop environment within the web browser, allowing maps to be organized within windows and manipulated in user-friendly ways. Additionally, the UC-IC platform was designed from the ground up to provide a social environment that enables the real-time transfer of applications and information to and from collaborators. This paper will show how the UC-IC platform allows for novel ways of interacting with maps containing real-time GIS data.

The rest of this paper is organized as follows. *Background* introduces the technology involved in displaying real-time data on the web and analyzes how the approach presented in this paper differs from existing solutions. *System Design* then discusses the unique client and server design of the proposed system. *Results* offers a look at two different deployments of the architecture: one with real-time data provided by running instances of Microsoft Flight Simulator, and the other using live SOS sensor data. Finally, *Conclusion* reflects on the contributions of this paper and proposes topics for future research.

Background

It was just over a decade ago that the Internet was limited to static content accessed using archaic web browsers. Despite these limitations, Geographic Information System (GIS) data quickly found its place on the Internet. Released in 1996 as a free service, MapQuest (9) made it possible for users to search for a location by name and to navigate the resulting map by using several buttons that surrounded the static map image (including buttons to select the zoom level). Users would see the entire webpage refresh for each navigation operation. Even with these inconveniences, there was something appealing about the interactivity and accessibility brought forth by the Internet that made it a good fit for geographic information.

As the Internet evolved to support more dynamic webpage content through the introduction of Rich Internet Application (RIA) technologies such as Asynchronous JavaScript and XML (AJAX), it came as no surprise that one of the first websites to make the most out of this new technology was a mapping site, namely Google Maps (6). Launched in 2004, Google Maps combined visually appealing maps with a very accessible user interface. It used AJAX to dynamically load sections of the map as the user dragged the map with the mouse cursor, a defining characteristic of what are now known as "slippy maps" (2).

While Google Maps offered many benefits (including that the service and developer tools were available at no charge), it also had some shortcomings. Its simplified interface, while pleasant to use, omitted support for the addition and selection of user-defined layers, such as layers based on the Web Map Service (WMS) standard defined by the OGC. The API available to developers required obtaining a key from Google, which was only functional on one domain and imposed numerous restrictions on how the map was to be used, including the following as described in the Google Maps Terms and Conditions:

> Except where you have been specifically licensed to do so by Google, you may not use Google Maps with any products, systems, or applications installed or otherwise connected to or in communication with vehicles, capable of vehicle navigation, positioning, dispatch, real time route guidance, fleet management or similar applications.(1)

Another limitation of the Google Maps API was that the API code itself could not be deployed on servers not owned by Google, meaning that developers had to rely on Google's uptime and availability. If developers wanted to demo their browser-based map application in an area without Internet access, they would be out of luck.

Several open source "slippy maps" emerged to address these drawbacks, with OpenLayers currently being the largest after absorbing many of the developers from the now-defunct MapBuilder project (8). OpenLayers has appeared in several academic papers (19)(17). There are no references to any "slippy map" being used for real-time data within a flexible collaborative environment, however.

Several papers have looked into the implications of real-time data delivery to a browser (20). In addition, online "WebOS" environments have existed for some time (3), although interaction with other users in these environments is usually limited to basic sharing of media, and they do not allow for full real-time collaboration of entire web-based applications and their data.

The publish/subscribe messaging model is a well known solution for real-time communication and is used in numerous examples, including as part of the OGC Sensor Alert Service (SAS) implementation of 52North. In this SAS architecture, both Publisher and Subscriber register themselves on the SAS server and communicate with each other using the open Extensible Messaging and Presence Protocol (XMPP) through a Multi-User Chat (MUC) channel (11). This paper, however, will present an approach that uses a registry service and other techniques to ensure that the architecture is scalable and robust enough for streaming data to a real-time social networking environment.

System Design

This section presents the client-side and server-side design that allows real-time GIS data to be delivered to a collaborative web-based environment.

Client-Side Design

The client-side user interface is a web-based implementation of the desktop metaphor, consisting of familiar windows, icons and menus that can be manipulated by using the mouse. The entire environment, however, is built on top of a collaborative platform that allows any window to be "sent" to another user. This is done simply by dragging a window onto an icon on the web-based desktop representing the friend who is to receive the application (and who must acknowledge a dialog to accept it). This sending process is unique in that the entire application logic, in addition to the current data within that application, is sent as part of the window. This is made possible by an advanced XML-based syntax and dynamic resource loading techniques (AJAX-Push) that go beyond the scope of this paper.

What is important to note, however, is that this concept of "sending" does not necessarily mean that the user doing the sending no longer retains the application. Rather, both users can have the same window open, and the inherent collaboration built into the system ensures that any actions performed within that application are automatically synchronized to the other user. For example, any text being typed into a text field will be communicated character-by-character to the other user's browser so that, as much as possible, both users always see the same application state. The environment was named "UC-IC" to highlight these collaborative characteristics.

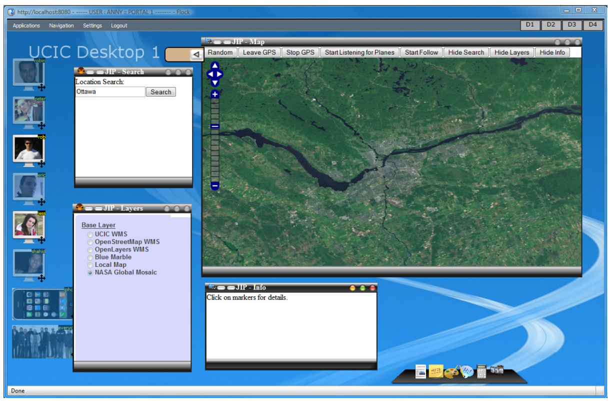


Figure 1: UC-IC environment with JIP Windows open.

Real-time collaboration on UC-IC is supported for applications programmed in, or able to communicate through, DHTML (AJAX). This includes Java Applets and Adobe Flash components, all of which can communicate through the DHTML-based UC-IC platform, offering a flexible environment for collaborative application development. Existing applications built on the UC-IC platform include a videoconferencing/chat application, a rich-text editor for live co-authoring, a collaborative video player, and a drawing application that can be reused for annotations on top of other applications.

The real-time nature of the platform makes it ideal for GIS applications dealing with sensor data. The GIS application implemented on the UC-IC platform was named Joint Intelligence Picture (JIP) and is shown in Figure 1. It consists of four different windows that communicate with each other through methods provided by the platform:

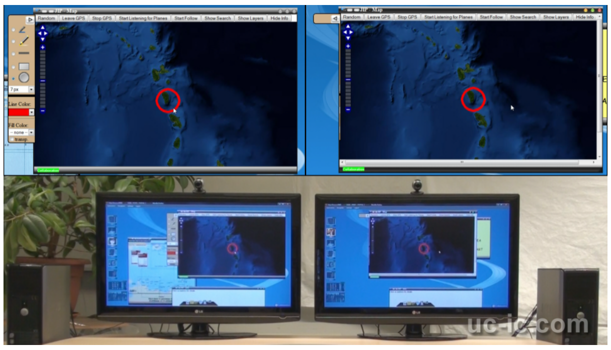


Figure 2: Two users collaborating on a JIP Map Window inside UC-IC.

JIP Map Window The Map Window contains an interactive "slippy map" generated by the OpenLayers API (10). It is the main window that appears when JIP is selected from the UC-IC applications list and it contains buttons for hiding and showing the other three windows that make up JIP. Inviting other users to collaborate on a JIP Map Window allows all of those users to see actions such as zooming, panning and drawing on the map. Figure 2 shows how a red circle drawn by one user appears on the collaborative Map Window of another user. A close-up of each screen is provided in the top half of the image. Other JIP Windows can also affect the contents of the Map Window; for example, using the Search Window can invoke the Map Window to show a specific location. In addition, markers may be displayed on the map based on real-time sensor data (for example, GPS coordinates of a moving vehicle).

JIP Search Window The Search Window contains a search box and list area where search results appear. Clicking on a search result updates the JIP Map Window. A custom geocoding solution provided by M3Data (4) is used to generate the results.

JIP Layers Window Part of the OpenLayers API, the Layers Window allows users to choose from a list of predefined map base layers. The list contains several custom map layers conforming to the OGC WMS standard and hosted on a local GeoServer (12) deployment, as well as free WMS layers from NASA (18) and OpenSteetMaps (15). In addition, the OpenLayers API supports loading layers from commercial services such as Google Maps, although these were avoided for reasons related to ease of deployment as mentioned in section *Background*. The Layers Window also contains checkboxes for showing and hiding the sensor data markers that are to appear on the map.

JIP Data Window The Data Window contains information based on user clicks in the Map Window. For example, a user can click on a barometric sensor marker on the map and see a real-time dial indicating barometric pressure in the Data Window. Like all windows in the UC-IC environment, the Data Window can be shared with other users on its own (for example, if a receiving user only needs to watch the live dial move and does not need the corresponding map).

Server-Side Design

In order to obtain real-time SOS data and display it within the JIP Map Window or Data Window, the web browser has to be in constant communication with the UC-IC server using AJAX. The Java Enterprise Edition (JEE) technology found in the open source JBoss Application Server (5) is used to make this possible. Of particular importance to the real-time architecture is JBoss' built-in Java Message Service (JMS) Server called JBoss Messaging.

JMS allows for asynchronous messaging based on the publish/subscribe messaging software pattern. The purpose of a JMS Server (sometimes also called a JMS Provider) is to route messages between JMS Clients, which can be either JMS Publishers or JMS Subscribers. JMS Publishers publish messages to a certain "topic" on the JMS Server, and JMS Subscribers subscribe to that topic to asynchronously receive the messages. The JBoss Application Server includes a JMS Server, while JMS Clients can be Java applications that use the JMS API.

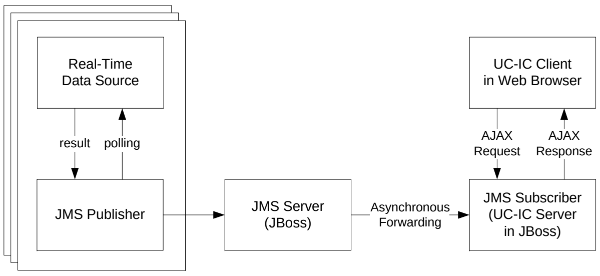


Figure 3: Basic real-time architecture using JMS.

A basic real-time architecture is shown in Figure 3. To initiate the transfer of real-time data, the client (the user in a web browser) must send an AJAX request that causes the UC-IC server (the JMS Subscriber) to subscribe to the real-time stream via the JMS Server. A JMS Publisher component is tasked with polling a real-time data source and receiving its response. The JMS Publisher then publishes the response to the JMS Server using a pre-arranged topic. The JMS Server then uses asynchronous messaging to forward the results to any JMS Subscribers who have subscribed to receive messages on that specific topic. Finally, the subscribed UC-IC Server sends the message to the client using AJAX-Push. The browser based client then parses the data and displays it within JIP.

This sequence of events can be seen in Figure 4. Note that, as was shown in Figure 3, multiple JMS Publishers can be sending real-time data to the JMS Server at the same time. Additionally, standard JMS methods exist for unsubscribing (based on an AJAX request from a user) and terminating the publishing process.

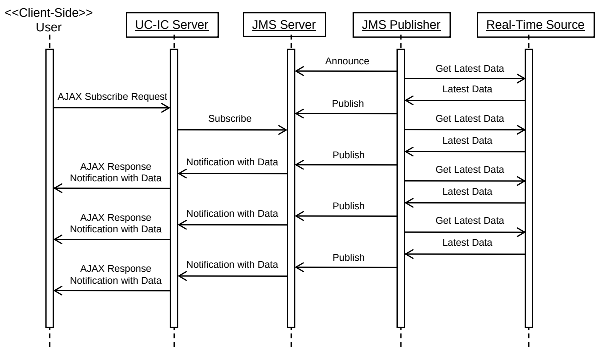


Figure 4: Event sequence for basic real-time data delivery to web browser.

While the above architecture would be sufficient for basic real-time data transmission, our final architecture features several enhancements that offer additional scalability and robustness required for real-time delivery to a social and collaborative platform like UC-IC. Having one centralized JMS Server, for example, is not ideal for real-time data delivery since the JMS Server could suffer an outage or become overloaded. A JNDI (Java Naming and Directory Interface) Registry Server is therefore introduced. The JNDI Registry Server stores a list of topics and the network address of the corresponding JMS Server where the topic is managed. JMS Publishers and Subscribers, which are pre-configured with the topics that they are to access, must first request the location of a JMS Server via the JNDI Registry Server. Once they obtain the location of the JMS Server, they can proceed as above.

By using a JNDI Registry Server, new JMS Servers can be added to the system with ease, allowing for much better scalability of the system as the number of topics increases. In addition, the support for redundancy of the system is increased since topics can be reassigned to different JMS Servers by changing the values stored by the JNDI Registry Server. By changing the location of the topics, JMS Clients can be dynamically assigned to the best JMS Server for optimal real-time data delivery.

The architecture robustness is further increased by keeping a list of alternate JNDI Registry Server locations in the JMS Servers and JMS Clients. The architecture can also support security through the use of SSL certificates, although security implications are beyond the scope of this paper.

Results

The design described in section *System Design* was applied to two different deployments: one with real-time data from Microsoft Flight Simulator, and the other using live sensor data based on the Sensor Observation Service (SOS) standard.

Flight Simulator

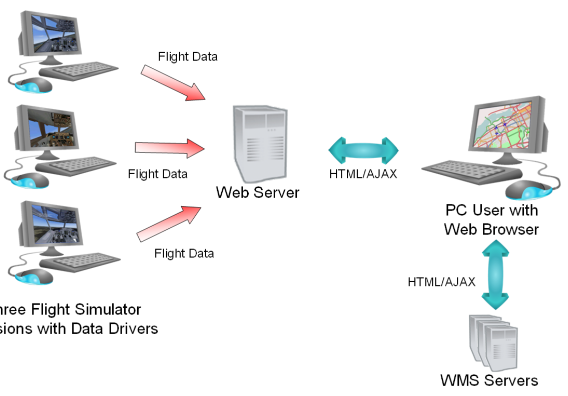
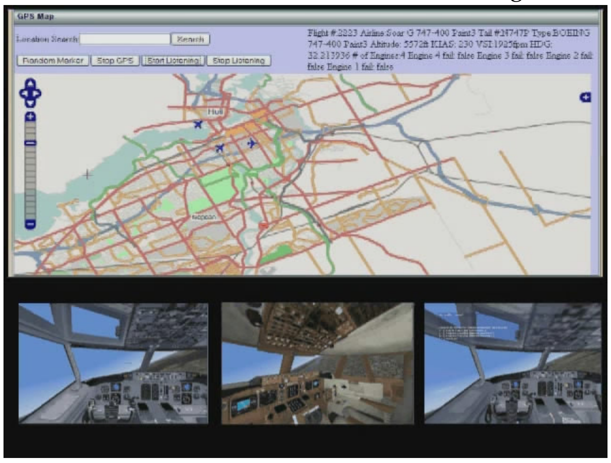


Figure 5: Flight Simulator test setup overview.

To test how the real-time architecture functions within a collaborative web-based environment, a controllable real-time data source was needed. Microsoft Flight Simulator 2004 was selected since the real-time aircraft data from inside the game could be accessed through a driver called FSUIPC (16). By setting up the JMS Publisher with the driver as the data source, XML data could be obtained for asynchronous sending to the UC-IC Server and display on the JIP client. A high-level view of this architecture is summarized in Figure 5.



Flight #2223 Airline:Soar G 747-400 Paint3 Tail #N747P Type:BOEING 747-400 Paint3 Altitude: 5572ft KIAS: 230 VSI:1925fpm HDG: 32.213936 # of Engines:4 Engine 4 fail: false Engine 3 fail: false Engine 2 fail: false Engine 1 fail: false

Figure 6: Real-time data from three different Microsoft Flight Simulator sessions displayed inside JIP.

The system was tested with three different Flight Simulator instances running on three different computers, each with their own JMS Publisher. This is shown in Figure 6. As the planes moved, a marker on the JIP client would move to show each plane's latest location. In addition, clicking a marker would populate the JIP Data Window with additional streaming information, such as the plane's current altitude, air speed and heading.

Although it was clear that a stream of data from the game was being received by the browser, the exact delay in the data was difficult to gauge. An additional feature was added that would change the plane marker color to red in the case of a simulated engine failure, which could be quickly activated from within Flight Simulator. The delay was observed to be less than two seconds when testing on a local network. The JIP Map Window was shared with five other UC-IC users, who could navigate and draw on the map while the planes were moving.

SOS Data

To test the system with more typical GIS sensor data, a standard SOS server was set up by hosting the open source 52North SOS Server component (13). The JMS Publisher component was tasked with polling the SOS Server using GetObservation requests. The JMS Publisher then received a response in the OGC Observation & Measurements (O&M) format. The JMS Publisher component then sent the O&M response to the JMS Server component, which would asynchronously forward it to the UC-IC Server, and the UC-IC Server would send it for parsing and display on the client using AJAX. In this case, an open source library called JFreeChart was used to display the data as a real-time dial (7).

Again, the real-time performance was very good with almost no noticeable delay, and the collaborative nature of the system made it easy to monitor and annotate the data as a group.

Conclusion

This paper introduced an architecture used to deliver real-time Flight Simulator and SOS sensor data to a social and collaborative UC-IC environment accessible from within a web browser. This was accomplished by using open source technologies, including OpenLayers to display the real-time data and JBoss to make asynchronous communication possible. The system was tested by using real-time data from Microsoft Flight Simulator and a locally-hosted SOS server using open source components from 52North. In both cases, the real-time data streamed smoothly to the JIP client application and was available for real-time collaboration with other users of UC-IC.

Future work will attempt to address limitations of JavaScript memory management techniques used by browsers, which can cause performance issues if the real-time data is left streaming for long periods of time. The SOS standard, for example, offers a less-verbose GetResult request which is more ideal for real-time data and would allow for more complex sets of real-time data to be delivered to the client. Although experiments with mobile devices have already been undertaken, the great breadth of mobile web browsers has given inconsistent results when attempting to load the DHTML-based UC-IC environment, and are worth exploring further. Finally, other standards such as the Sensor Alert Service could be implemented to communicate scenarios like the plane engine failure.

Rabih Dagher, Cristian Gadea, Bogdan Ionescu, Dan Ionescu and Robin Tropper,
NCCT Laboratory
University of Ottawa
161 Louis Pasteur Room B-306
Ottawa ON K1N-6N5
CANADA
rdagher AT ncct.uottawa.ca
cgadea AT ncct.uottawa.ca
bogdan AT ncct.uottawa.ca
dan AT ncct.uottawa.ca
rtropper AT ncct.uottawa.ca

Bibliography

[1] Google Maps Terms and Conditions. URL http://www.google.com/intl/en_ALL/help/terms_local.html. [Accessed: July 2009]

[2] Definition: Slippy Map (2005). URL http://fantomplanet.wordpress.com/*2005/06/23/definition-slippy-map/. [Accessed: July 2009]

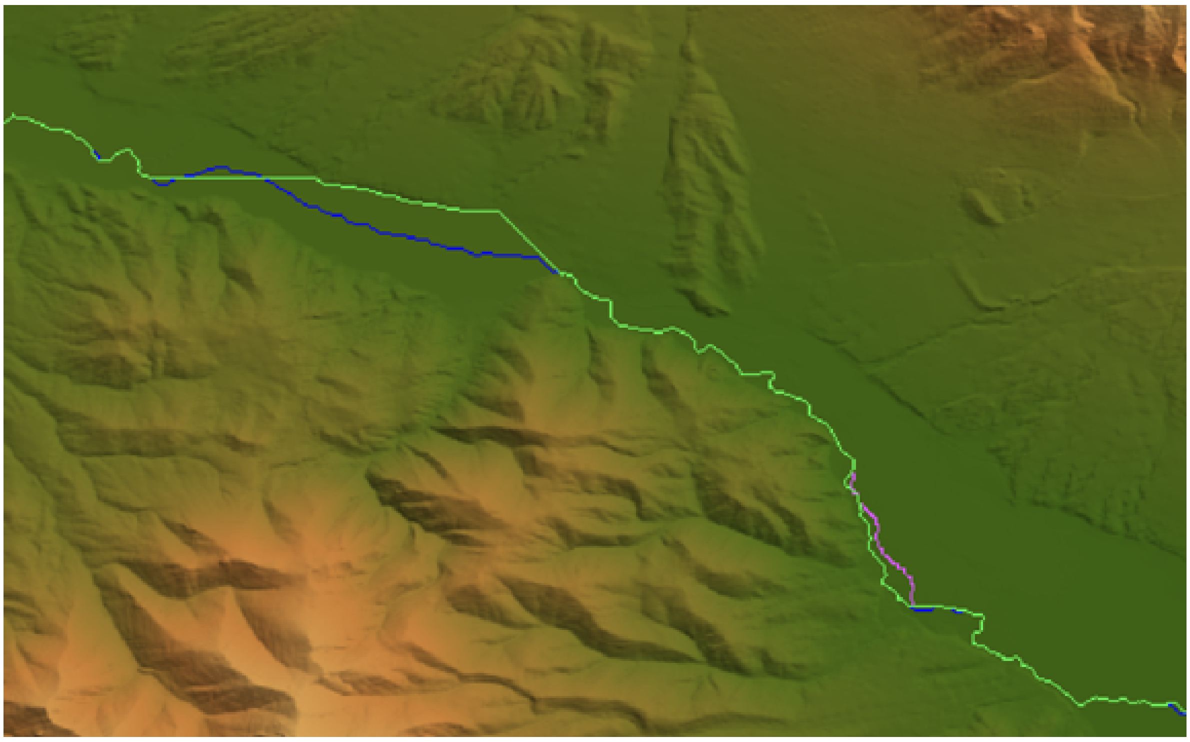


Figure 13: The results of the three models overlaid above the original. Green is the top layer and is the result of the ArcGIS ModelBuilder, Purple the result of the Sextante Modeler and blue the result of the MapWindow Modeler.

Discussion and Conclusions

The MapWindow GIS Modeler is a versatile modeling environment, which can handle many different data types. It executes models in similar time to other GIS modeling environments and faster than other open source ones. Due to its modular and extensible architecture it can use tools of many different designs. The design flexibility not only allows tools to function in a wide variety of different ways, but it allows tools and their associated parameters to be generated from any number of sources. Its ease of use for end users and developers, as well as its integration with MapWindow GIS 6 and MapWindow GIS 4, ensures that the widest range of users will have access to the program. By building on the successful design of previous generations of MapWindow GIS, the MapWindow Modeler benefits from all of the development expertise, keeping the designs that were the most effective while eliminating some of the more constrictive problems. It is one more tool available to both developers looking to create new modeling tools and end users wishing to create models with such tools.

Brian Marchionni & Daniel Ames,
Idaho State University
marcbria@isu.edu
dan.ames@isu.edu

Bibliography

[1] D. Ames. *MapWinGIS Reference Manual: A function guide for the free MapWindow GIS ActiveX map component*. Lulu.com, Morrisville, North Carolina, 2007.

[2] D. Ames, C. Michaelis, and T. Dunsford. Introducing the mapwindow gis project. *The Journal of the Open Source Geospatial Foundation*, 2:13–16, 2007.

[3] S. Greenberg. Toolkits and interface creativity. *Multimedia Tools and Applications*, 32(2):139–159, 2007.

[4] J. B. Gregersen, P. J. A. Gijsbers, and S. J. P. Westen. Openmi: Open modelling interface. *Journal of Hydroinformatics*, 9(3):175–191, 2007.

[5] L. Liquori and A. Spiwack. Extending feathertrait java with interfaces. *Theoretical Computer Science*, 398(1-3):243–260, 2008.

[6] T. Maxwell. A paris-model approach to modular simulation. *Environmental Modelling & Software*, 14(6):511–517, 1999.

[7] V. Olaya and J. C. Gimenez. *SEXTANTE: a gvSIG-based platform for geographical analysis*. Free and Open Source Software for Geospatial, Victoria, Canada, 2007.

[8] K. L. Verdin and J. P. Verdin. A topological system for delineation and codification of the earth's river basins. *Journal of Hydrology*, 218(1-2):1–12, 1999.

[9] Y. Xie and D. G. Brown. Simulation in spatial analysis and modeling. *Computers, Environment and Urban Systems*, 31(3):229–231, 2007.

Imprint

Editor in Chief:
Tyler Mitchell - tmitchell AT osgeo.org

Assistant Editor:
Landon Blake

Section Editors & Review Team:
Eli Adam
Daniel Ames
Dr. Franz-Josef Behr
Jason Fournier
Dimitris Kotzinos
Scott Mitchell
Barry Rowlingson
Jorge Sanz
Micha Silver
Dr. Rafal Wawer
Zachary Woolard

Acknowledgements
Daniel Holt, LaTeX magic & layout support
Various reviewers & writers

The *OSGeo Journal* is a publication of the *OSGeo Foundation*. The base of this journal, the LaTeX 2ε style source has been kindly provided by the GRASS and R News editorial boards.

Journal online: http://www.osgeo.org/journal

OSGeo Homepage: http://www.osgeo.org

Mail contact through OSGeo, PO Box 4844, Williams Lake, British Columbia, Canada, V2G 2V8

9 771994 189002 00

ISSN 1994-1897