

INTEGRATION OF JAVA/JAVA3D-BASED OCEANOGRAPHIC ANALYSIS TOOLS WITH GIS

Tiffany C. Vance¹, Christopher W. Moore² and Nazila Merati²

¹NOAA/National Marine Fisheries Service, Seattle, WA

²Joint Institute for the Study of the Atmosphere and Ocean, University of Washington, Seattle

ABSTRACT

Ideally, scientists should be able to format, explore, analyze, and visualize GIS-based data in a simple, powerful, and fast application that would seamlessly integrate georeferenced data from a variety of data sources into a powerful intuitive visualization. Virtual reality has made visualization easily accessible to many users using desktop tools but has limitations in scope and analysis. In this project we used a combination of Java/Java3D, the Visualization Toolkit, and the recently introduced ESRI ArcGIS Engine product to create a prototype of a tool to calculate a number of parameters for an oceanographic module to be included in a scientific GIS. We combined the spatial tools exposed through ArcGIS Engine with the analytical capabilities of algorithms written in Java and the complex visualization capabilities of Java3D. Modules from each of these technologies could be combined to create innovative tools to allow users to import data, perform spatial and scientific analyses and output the results as visualizations for further examination.

1. INTRODUCTION

Geographic information systems (GIS) provide a high level of functionality for spatial analyses, but are not yet able to provide extended functionality needed to create a truly "scientific GIS". Examples of the functionality that is lacking include time series analyses, calculation of the volume of the overlap between two volumes - for example between a school of fish and a prey field - or calculation of the intersection of a vector path with a volume - for example the route of a marine mammal through a pool of cold water. Other functions might include the ability to specify a slice through a three-dimensional lattice of model output data and to make various analyses along that slice.

Java can be used to program scientific calculations and analyses, but it isn't inherently spatial. Datasets can have a spatial component, but Java treats this as it would any type of coordinate system. Topology, or the

spatial relationship between objects, is not stored with data. Functions such as map projections, slope calculations and spatial intersections are not native to the language. However, Java is easily extensible and functions written in other languages can be integrated.

VRML provides the ability to visualize scientific data and to allow the user to interact with the data by rotating, zooming and panning, but you cannot easily query VRML objects. Ideally one would be able to point at a three dimensional location in a VRML view and return a reference to the object that lists the analytical methods that act on the data, and allows the user to execute these methods. VRML scene navigation generally requires that a separate VRML plug-in application be installed on the client system. From the user's perspective, plug-ins are poorly integrated into the browser environment, and the browser itself may be superfluous. Additionally, user plug-ins may only be available for the more popular client platforms. Recent developments in Java3D extend the functionality of VRML and answer a number of the limitations mentioned.

The Java tools described herein were designed using a framework approach that enables them to be integrated into an application (OceanGIS), or interfaced with major off-the-shelf software products (e.g., ESRI ArcEngine). We aimed to develop these tools for a broad range of potential users. This project focuses on tools that convert non-spatial data into GIS compatible data, expedite the transfer of spatial data to coastal professionals and emergency managers, and enhance analyses used for disaster preparedness and response activities. As the tools we develop can be deployed without a full ArcGIS license, we hope to make them widely available to better integrate field activities during disaster responses.

2. TECHNOLOGY

In this project, we use the flexibility of Java to integrate GIS functionality with Java3D (and Java-wrapped OpenGL) visualization capabilities. Specifically we are using a combination of Java/Java3D and the recently introduced ArcEngine product to create a prototype of a scientific GIS. We combine the spatial tools exposed through the ArcEngine Java API with the analytical capabilities of algorithms written in Java with the

* *Corresponding author address:* Tiffany C. Vance, NOAA/National Marine Fisheries Service, 7600 Sandpoint Way NE, Seattle, WA 98115; e-mail: Tiffany.C.Vance@noaa.gov

complex visualization capabilities of Java3D. Modules from each of these technologies will be combined to create innovative tools to allow users to import georeferenced data, make spatial selections, perform spatial and scientific analyses and output the results as visualizations for further examination. Use of the ArcIMS Java Connector will allow these modules to be implemented in ArcIMS sites for web-based analysis.

ArcGIS Engine is an ESRI developer product for creating and deploying ArcGIS solutions. It is a simple API-neutral cross-platform development environment for ArcObjects - the C++ component technology framework used to build ArcGIS. ArcObjects are the core of the ArcGIS functionality and include tools such as overlay - union, intersect; proximity - buffer, point distance; surface analysis - aspect, hillshade, slope; and data conversion - shapefile, coverage and DEM to geodatabase. ArcEngine's object library makes full GIS functionality available through fine- and coarse-grained components that can be used in Java and other environments. Using ArcEngine, one can build solution and deploy them to users without requiring the ArcGIS Desktop applications (ArcMap, ArcCatalog) to be present on the same machine. It supports all the standard development environments, including Java, C++, and .NET, and all the major operating systems. In addition, one can embed some of the functionality available in the ArcGIS extensions. This product is a developer kit as well as deployment package for ArcObjects technology. Using ArcEngine we will integrate GIS functionality into an application with the data being available for calculations in non-GIS components. We will also be able to make these tools available to ArcIMS sites. ArcIMS has a limited set of spatial capabilities but it is capable of interfacing with the Java3D API via the Java Connector. This will allow the ArcIMS community to utilize tools built with this project.

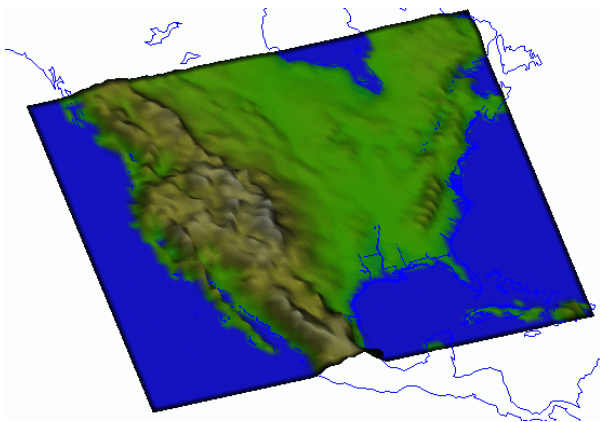


Figure 1. Visualization of topography made using IDV
[<http://www.unidata.ucar.edu/content/software/IDV/gallery>]

Java allows us to make a variety of scientific calculations on the data and to provide the results back both to the GIS component and to a Java3D based visualization component. We are able to take advantage of a number of Java utilities such as Unidata IDV (Figure 1), OceanShare, ncBrowse, SGT toolkit, and TimeSeries applet. The ArcIMS Java Connector could be used to produce a map coordinates base that would allow data retrieval from a DODS server, and subsequent plotting with tools designed for interaction with gridded fields.

The Java3D API is an application programming interface used for writing stand-alone three-dimensional graphics applications or Web-based 3D applets. It gives developers high level constructs for creating and manipulating 3D geometry and tools for constructing the structures used in rendering that geometry. With Java3D API constructs, application developers can describe very large virtual worlds, which, in turn, are efficiently rendered by the Java3D API. The Java3D API extension is designed as a high-level platform-independent 3D graphics programming API and is amenable to very high performance implementations across a range of platforms. To optimize rendering, Java3D implementations are layered to take advantage of the native, low-level graphics API available on a given system. In particular, Java3D API implementations are available that utilize OpenGL, Direct3D, and QuickDraw3D. This means that Java3D rendering will be accelerated across the same wide range of systems that are supported by these low-level APIs. The Java3D API is aimed at a wide range of 3D-capable hardware and software platforms, from low cost PC game cards and software renderers, through mid-range workstations, all the way up to very high-performance, specialized, 3D image generators. Support for run-time loaders was included to allow Java3D to handle a wide variety of file formats such as interchange formats, VRML 1.0, and VRML 2.0.

We also explored the use of a second 3D API called the Visualization Toolkit (www.kitware.com). Like Java3D, VTK is a cross-platform 3D application programming interface built upon, and independent of, the native rendering library (OpenGL, etc). It exposes a Java bindings (as well as Tcl, and Python). It is written in C++, uses the object oriented modeling approach of Rumbaugh, et al., and includes similar scene-graph, lighting models, and graphic primitives as Java3D. What VTK does that Java3D doesn't (yet) do is boolean operations on 3D volumes (intersection, union), volume rendering, filtering, including convolution, FFT, Gaussian, Sobel filters, permutation, high- and low-pass Butterworth filters, and divergence and gradient calculation. The VTK data model allows for fast topology traversal, making these filters very

fast, and allows for rapid mesh decimation. VTK also offers powerful 3D probe "widgets" that allow easy interaction with the data, and has methods to utilize parallel architecture through the Message Passing Interface (MPI).

3. APPLICATIONS

Combining these technologies, we will be creating applications for the NOAA nowCoast project and for the NOAA National Marine Fisheries Service (NMFS). The nowCoast project is an Office of Coast Survey effort to provide forecast model developers and the coastal community with centralized access to real-time physical, meteorological, oceanographic, river, and air/water quality information. The Web portal (Figure 2) also provides NOAA forecasts for major estuaries, seaports, and adjacent coastal regions as well as the Great Lakes. The application this project develops will serve 3D rendered objects of model data. It will enhance nowCoast by the addition of a number of 3D visualization and spatial analysis tools. These will include creating 2D and 3D plots from a polyline, creating 2D velocity plots from point input (columnar plot) and a capability for viewing model data in 3D. A

second application, for the NOAA/NMFS Alaska Fisheries Science Center, will allow scientists to calculate a variety of statistics and measure about the intersection of vector and volumetric objects with volumes in 3D. The objects could include marine mammal tracklines and schools of fish and the 3D volumes might include schools of prey and water masses such as cold pools. These types of volume on volume intersections could be generalized to a number of coastal and offshore applications. They will also serve as templates for many other tasks.

4. INITIAL RESULTS

A test application using the project framework was put together to view high-resolution global topography/bathymetry, ocean model output, and standard ocean hydrographic data. For this application (dubbed OceanGIS) we created a number of Java objects that wrap data and methods for acting on the data. For bathymetry data, this was just the ability to exaggerate the vertical coordinate and to decimate and smooth the resulting mesh. But for more complicated data like hydrographic surveys, the objects allow for the dynamic creation of such typical oceanographic

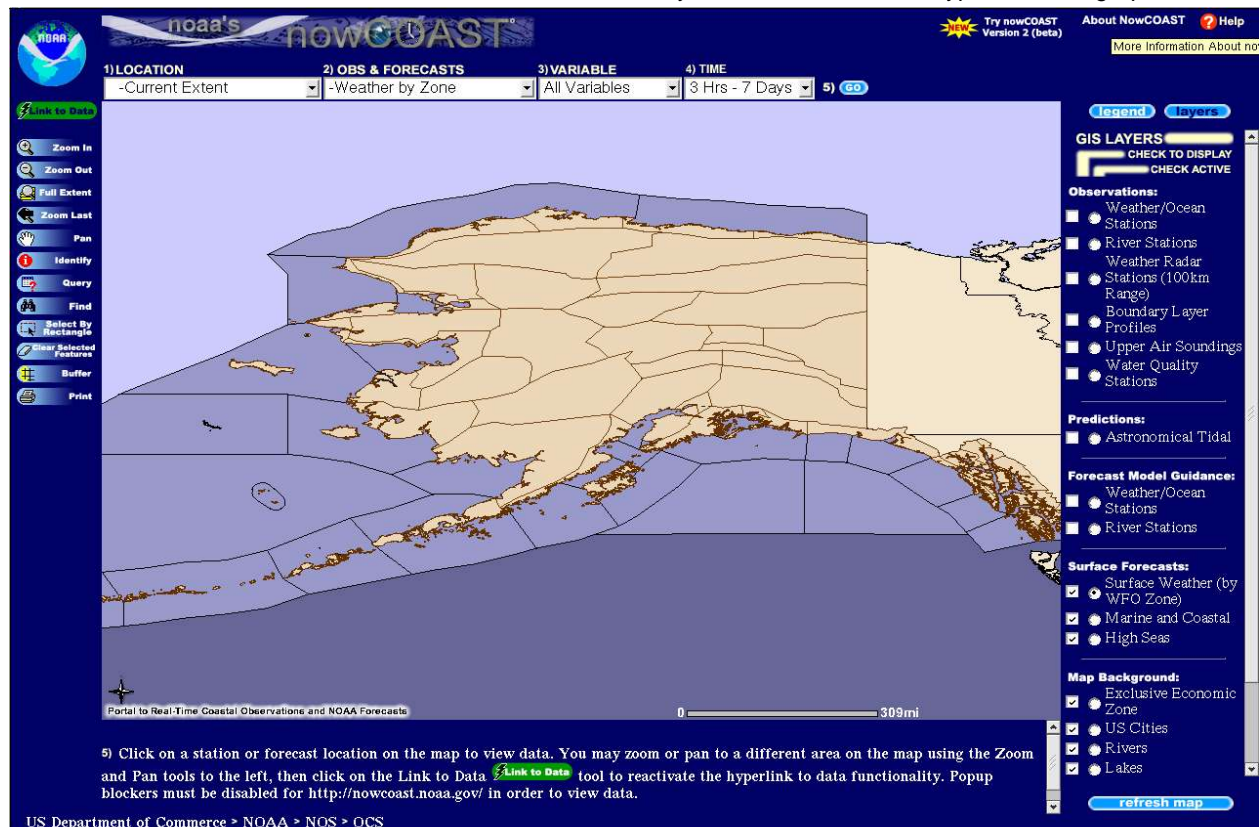


Figure 2. The nowCoast web portal utilizing ArcIMS [<http://nowcoast.noaa.gov/viewer.htm>]

parameters as dynamic height, mixed-layer depth, and geostrophic flow. For model output, variables such as salinity can be shown as an isosurface or vector fields such as water velocity can be shown as a plane of 3D vectors that the user can move to "probe" the data. The user can seed the flow field with lagrangian floats that are time-stepped to show particle paths.

In Figure 3, bathymetry from the Smith and Sandwell dataset are selected in the Aleutian Island change, and the vertical dimension is exaggerated to show the Aleutian trench. A CTD survey just north of the archipelago is imported in shapefile format, and mixed-layer depth is calculated using a kriging method. Note the data-object icon on the left panel, exposing tool methods to the user - in this case a hydrographic data object was clicked.

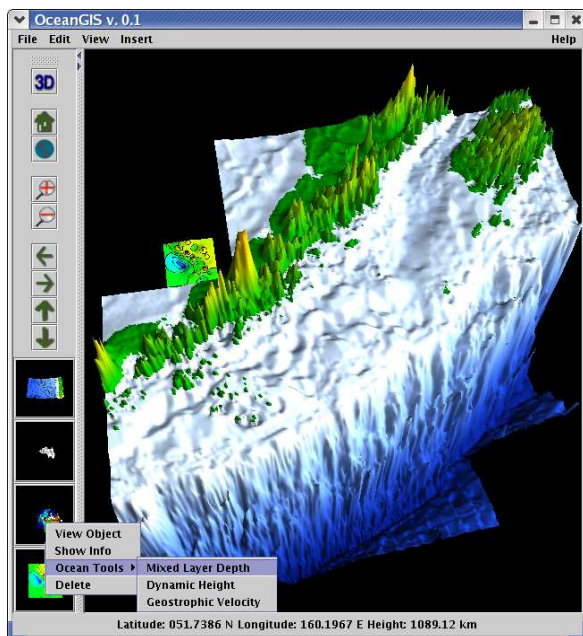


Figure 3. OceanGIS with data object icons on the left showing tools available for hydrographic data [<http://www.pmel.noaa.gov/vrl/OceanGIS>]

Figure 4 shows an even higher-resolution bathymetric data set (multi-beam), as well as model output of a hydrothermal vent plume. Dissolved aluminum concentration is shown in silver, and a plane probe widget slices through the data. The user can re-size the widget or push it through the volume to show concentration at that location, or to calculate volume of the plume.

5. CONCLUSION

While none of the various technologies we have used is complete in and of itself, the linking of GIS, Java, Java3D and VTK provides a powerful mechanism to

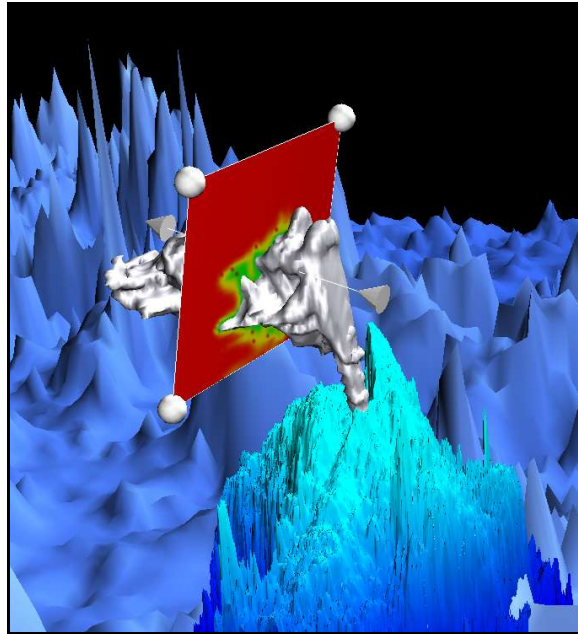


Figure 4. OceanGIS 3D probe widget probing hydrothermal vent plume model output.

create the beginnings of a "scientific GIS". Once the technologies have been linked, the form of the final tools deployed will be dependent upon the users needs and the relative costs of the GIS portions of the technologies. Because we are creating scientific tools rather than a commercial product, license costs may have to play a large role in our choice of which GIS tools to use.

6. ACKNOWLEDGEMENT

This publication was supported by the Joint Institute for the Study of the Atmosphere and Ocean (JISAO) under NOAA Cooperative Agreement #NA17RJ1232, Contribution #1102. PMEL contribution 2769. The views expressed herein are those of the author(s) and do not necessarily reflect the views of NOAA or any of its sub-agencies. This work was funded by NOAA's HPCC program.

7. REFERENCES

PMEL scientific visualization website:
<http://www.pmel.noaa.gov/vrl/OceanGIS>

VTK website:
<http://www.kitware.com/VTK>

Java3D website:
<http://java.sun.com/products/java-media/3D>