

# INTERACTIVE VISUALIZATION OF REGIONAL OCEAN MODELING SYSTEM

Yuzhong Shen

Department of Electrical and Computer Engineering  
Old Dominion University  
Norfolk, VA 23529, USA  
email: yshen@odu.edu

Jay A. Austin

Large Lakes Observatory  
University of Minnesota  
Duluth, MN 55812, USA  
email: jaustin@d.umn.edu

Jessica R. Crouch

Department of Computer Science  
Old Dominion University  
Norfolk, VA 23529, USA  
email: jrcrouch@odu.edu

Michael S. Dinniman

Center for Coastal and Physical Oceanography  
Old Dominion University  
Norfolk, VA 23529, USA  
email: msd@ccpo.odu.edu

## ABSTRACT

This paper presents techniques for interactive visualization of the Regional Ocean Modeling System (ROMS), a free-surface, terrain-following coordinate ocean circulation model. We have designed an interactive visualization program that allows a casual user to control the forcing conditions applied to ROMS using a graphical user interface and observe the results in real-time. We modified portions of ROMS to facilitate user interaction with a Chesapeake Bay model. In order to achieve real-time interactive visualization, a low-resolution computational grid of the Chesapeake Bay was employed and a variety of techniques were utilized to achieve appealing visualizations similar to those created from high-resolution models. Texture mapping and alpha-channel blending were used to create a smooth boundary of the bay and filtering applied to adjacent grid points eliminates aliasing. The integrated simulation and visualization environment allows a user to control wind speed and direction along with the rate of flow from the rivers that feed the bay. It also provides visualizations of water height, velocity, salinity across horizontal and vertical planes, and particle tracking. Many interesting oceanographic phenomena can be demonstrated with the interactive program, making it a tool that facilitates oceanographic teaching and research.

## KEY WORDS

Interactive, visualization, texture mapping, filtering, antialiasing, renderbin.

## 1 Introduction

Oceans cover more than two-thirds of the Earth's surface, and have fascinated humans since ancient times. Modern oceanography studies many aspects of the oceans. Biological oceanography studies the plants and animals in the ocean and their ecological interaction; chemical oceanography is concerned with the chemistry of the ocean; geolog-

ical oceanography explores the geology of the ocean floor including plate tectonics; and physical oceanography investigates the ocean's physical attributes, such as temperature, salinity, and circulation. With the aid of digital computers, an array of numerical ocean models have been developed to facilitate research in one or several subfields of oceanography [1–4]. Different ocean models can be loosely characterized by their approaches to spatial discretization and vertical coordinate treatment. Among them the Regional Ocean Model System (ROMS) [4] is a free-surface, terrain-following, primitive equations ocean model widely used by the scientific community for a diverse range of applications [5]. ROMS includes accurate and efficient physical and numerical algorithms and several coupled models for atmosphere and ocean, biogeochemical and ecosystem response.

Numerical ocean models such as ROMS produce large amount of data depicting various properties of the ocean, e.g., water temperature, flow velocity, and water density. Interpretation of simulation results relies heavily on various scientific visualization methods [6–10]. Scientific visualization [8] helps researchers gain insights into simulation or experiment results by creating various visual representations of the data. Effective and efficient visualization of ocean features, as ocean modeling itself, is a very active area of research. Numerical ocean models are typically run in an asynchronous fashion, with the investigator first setting up the conditions for the model, then running the model, and finally visualizing and interpreting the results. Both general and specialized visualization software are utilized in oceanography visualization. General visualization software such as MATLAB's visualization toolkit is used by oceanographers to carry out common visualization tasks. Specialized visualization software packages have also been developed to provide more advanced visualization capabilities for specific ocean models. For instance, The NSF Engineering Research Center at Mississippi State University developed CThru, a soft-

ware package for visualization of the Sea of Japan in the Naval Research Laboratory's Naval Layered Ocean Model (NLOM) model [11]. Although both the general and specialized visualization software packages provide users with a selection of several visualization methods, they are typically run as a post-processing step in ocean modeling and simulation. Due to their high computational complexity, ocean models usually require high-performance computing resources and the simulation takes a long time. Second, integration of model computation and visualization software requires researchers to have in-depth knowledge of both fields, which is rarely the case. Thus the most common means of sharing information between model computation and visualization software is through files, such as the netCDF format [12].

With separation of model computation and result visualization, oceanographers have to invoke two sets of tools, which imposes considerable inconvenience. An integrated interactive computing environment that allows users to set ocean model parameters, run the computation, and observe the visualization would benefit oceanography researchers greatly. One early research effort on interactive visualization of ocean circulation model was done by researchers at the NSF Engineering Research Center at Mississippi State University [7]. They ran the NLOM ocean model [2] on a Cray C90 supercomputer and carried out visualization on a Silicon Graphics workstation. The visualization also included a user interface through which the user could change some ocean model parameters. The visualization module then passed the parameter changes back to the Cray C90 supercomputer. The communications between the Cray C90 and the Silicon Graphics workstation was implemented using the Message Passing Interface (MPI) [13], a computer communications protocol that is the de facto standard for communication among the nodes of distributed memory systems. While this approach [7] was suitable for big computing centers, the usage of a supercomputer and expensive Silicon Graphics workstation prevented it from being used in a classroom or by users who only have access to personal computers.

In the last decade, both general purpose central processing units (CPUs) and graphics hardware (specifically, graphic processing units or GPUs) have undergone dramatic advances. Dual-core (or multiple-core) CPUs and powerful GPUs are now commonplace on personal computers and game consoles. It is advantageous to make use of the computing power of the latest VLSI technology for interactive simulation and visualization of ocean models. In our research, we developed a unified graphical user interface for both ocean model computation and visualization, and we run the ocean model and visualization processes on the same computer. Such design and implementation present several advantages. The implementation of interactive simulation and visualization on a single computer makes it an ideal tool for teaching oceanography, allowing either an instructor to demonstrate a particular phenomenon, or students to engage in active learning by using

the model themselves. Members of the larger community can also benefit from access to interactive visualization and simulation of numerical ocean models. For example, government and business planning officials and citizens concerned about environmental issues are all interested in understanding the impact that human activities could have on the environment. In addition, professional oceanographers can take advantages of such interactive tools that hide the intricate details of complex ocean models and save them time in model setup and result visualization.

The interactive visualization and simulation software presented in this paper was designed to make some of the capabilities of Regional Ocean Modeling System accessible to non-expert members of the community. The goal was to create a visualization and simulation program that would be both scientifically sophisticated and user-friendly. The design and implementation of this program involved re-engineering ROMS to run interactively instead of in batch mode, creating a new user-friendly graphical user interface, and integrating real-time 3D data visualization routines. The Regional Ocean Modeling system was applied to Chesapeake Bay, the largest estuary in the United States, to demonstrate the utility of the program. The resulting simulation and visualization allows a user to control wind speed and direction along with the rate of flow from the rivers that feed the bay. The program responds to changing conditions, takes tidal effects into account, and provides visualizations of water height, velocity, and salinity across the bays surface and in cross-section. The remainder of this paper is organized as follows. Section 2 describes the software architecture and visualization design of the interactive visualization and simulation program. Section 3 presents visualization results of the program. Section 4 concludes this paper and discusses possibilities for extending the current program.

## 2 Software Architecture and Visualization Design

### 2.1 Software Architecture

The interactive visualization program (Fig. 1) consists of three modules: the numerical ocean model, the graphical user interface, and the visualization module. The graphical user interface contains the main event loop that controls the program execution and flow of data at the highest level. It receives simulation and visualization parameters in the form of user inputs, passes the parameters to the numerical ocean model and visualization module, and updates the simulation results generated by the numerical model that are used by the visualization module. Description of each of the simulation components follows.

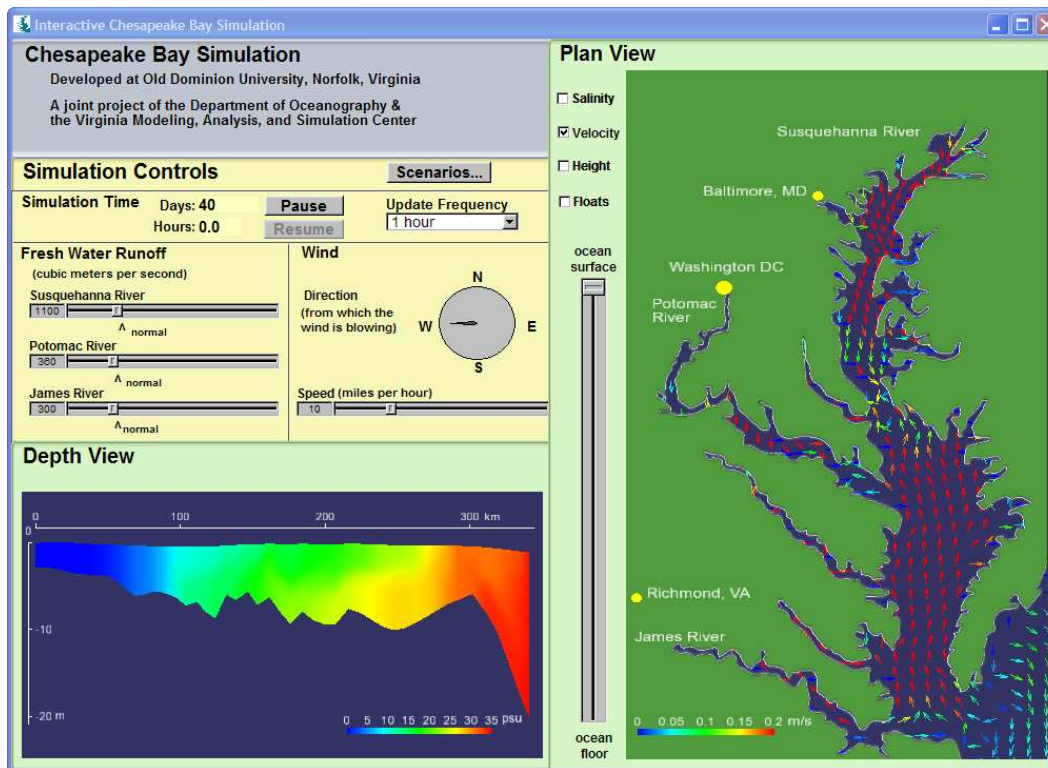


Figure 1. Interactive visualization of Regional Ocean Model System applied to the Chesapeake Bay

### 2.1.1 Numerical Ocean Model

The Regional Ocean Modeling System (ROMS) is a free-surface, terrain-following, primitive equations ocean model [5]. The terrain-following coordinates (sigma coordinates) allows the topography of the ocean floor and coast to be directly incorporated in a model. As input, ROMS requires parameters for the computational grid and a set of forcing conditions. Forcing conditions can include time and location dependent values for variables such as wind speed and direction, atmospheric temperature, precipitation, river flows, tides, and pollution sources. During the solution phase ROMS computes value of water depth, all three components of velocity, temperature and salinity throughout the three-dimensional volume of a model at a number of time steps. The source code of ROMS is written in Fortran 90 and consists of more than 340,000 lines. In order to produce compact, efficient executables from a large code base, ROMS makes extensive use of compiler directives to exclude portions of the code that are unnecessary for specific types of input models. This code structure makes editing and recompilation of the code necessary for each model variation. Furthermore, ROMS is designed to run in batch mode through input and output files and interactive adjustment to simulation variables is not supported. Therefore, ROMS provides researchers with low-level control and great modeling flexibility but lacks the user-friendly facilities needed in software intended for novice users. The requisite level of user sophistication pre-

cludes ROMS use by most students and casual users.

Modifications to ROMS were necessary in order to make its powerful numerical modeling capabilities more accessible. The outermost loop in the ROMS source code advances the simulation time and calls routines to set up and solve equations for the current time step. The primary modification to ROMS was the removal of this time-stepping loop and the addition of new top level functions to manage simulation time increments, update the forcing conditions at the beginning of each update, and pass the computed solution data to the visualization module at the end of each update. Originally written as a standalone executable program, the modified ROMS code was compiled into a linkable library called iROMS (interactive ROMS).

To apply iROMS to the Chesapeake Bay, a computational grid was defined to represent the entire Chesapeake Bay, stretching approximately 200 miles from Havre de Grace, Maryland, to Cape Henry, Virginia. Portions of several rivers feeding the Bay are represented in the grid, including the Susquehanna, Potomac, James, Rappahannock, and York Rivers. The dimensions of a computational grid are an important factor in the computational complexity of the iROMS solution algorithm. The goal for the Chesapeake Bay grid was to be small enough that the simulation would complete a solution step in less than a second while running on a PC and yet be large enough to demonstrate interesting effects in the Bay. The chosen grid dimensions are  $20 \times 50 \times 10$  (east-west  $\times$  north-south  $\times$  vertical layers)

as shown in Fig. 2(a). iROMS was configured by enabling the algorithms for computing salinity levels, tracking floating objects, using a K-profile parameterization for vertical mixing [14], controlling tides at the open boundary, and managing point sources of momentum, temperature, and salinity.

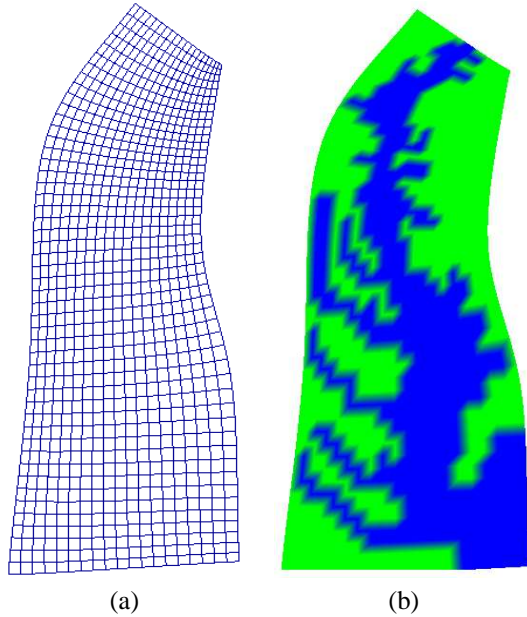


Figure 2. Grid configuration of ROMS on Chesapeake Bay. (a) The top layer of a  $20 \times 50 \times 10$  (east-west  $\times$  north-south  $\times$  vertical layers) grid for Chesapeake Bay. (b) The bay is shown in blue and the land around the bay is shown in green.

### 2.1.2 Graphical User Interface

A user interface was implemented using the Fast Light Toolkit (FLTK) [15]. The user interface code contains the programs main event loop that controls the simulations execution and flow of data at the highest level. Whenever a user changes the setting of one of the on-screen controls, the event loop receives notification of the change and updates the parameters passed to iROMS at the beginning of the next solution step. Calls to iROMS are implemented by spawning new threads. The multi-threaded nature of the program enables the user interface to provide smooth interactivity without blocking while waiting for iROMS to finish computing a solution step. Another benefit of the multi-threaded implementation is that the simulation can take advantage of a dual processor PC by continuously running iROMS calculations on one processor while using the other processor for interface and graphics functions. The interface provides two categories of controls: those that affect the numerical model and those that affect the visualization of computed results. Controls affecting the numerical model include a selection box for the update rate, sliders to

set river flow rates, and a slider and dial for setting the wind speed and direction. The interface displays the simulation time, updating it after each solution time step. An update frequency in the range of 1 to 24 hours can be selected, influencing the degree of variability due to tidal effects a user will notice. The rate of flow can be set for the three largest rivers feeding the Bay: the Susquehanna, Potomac, and James Rivers. Users may select a single wind speed and direction. Controls affecting the visualization of the computed results include check boxes to select a particular variable for visualization and a slider to select the depth of the grid layer to visualize. Either the water's salinity, velocity, height level, or the position of floating markers can be selected for visualization. Since the computed results are three dimensional, plotting data on a map of the Bay requires a two-dimensional subset of the data be selected. The depth slider allows users to select any of the grid's 10 vertical layers for visualization. The user interface is shown in Fig. 1.

### 2.1.3 Visualization Module

Visualization routines were written using OpenSceneGraph [16]. Plan view and depth view windows provide orthogonal visualizations of the Bay. The plan view displays a map of the Bay and surrounding coastal areas, complete with rivers and nearby cities. The following types of visualizations can be selected for the plan view. Water height is a scalar quantity represented by applying color mapping to the Bay. With a 1 hour update rate, tidal variations in water height are evident using this visualization. The water height visualization is displayed in Fig. 3(a). Salinity is another scalar quantity represented via color mapping. Variations in fresh water flow from the rivers feeding the Bay cause pronounced changes in salinity. Salinity visualization is shown in Fig. 3(b). Velocity vectors of unit length are drawn on the Bay to show the direction of water flow. The vectors are color mapped to indicate the magnitude of the velocity. Changes in the wind tend to affect water velocity. This visualization is shown in Fig. 1. Particle tracking allows visualization of water flow over time. Five floating virtual particles are dropped onto the surface of the Bay every 20 simulation days. The particle positions are tracked in iROMS and plotted at each time step. This visualization shows how winds, rivers, and tides can affect the path of floating objects over a multi-week time scale. Particle markers can be seen in Fig. 3(c). The depth view window provides a visualization of a cross-section of the bay that follows a center channel running roughly north to south. The bay's geometry in this view accurately represents the topography of the floor and the water's time and location dependent height variations. The salinity distribution on the cross-section is displayed via color mapping, as shown in the lower left corner of Fig. 1. The details of visualization design and implementation are discussed in the following section.

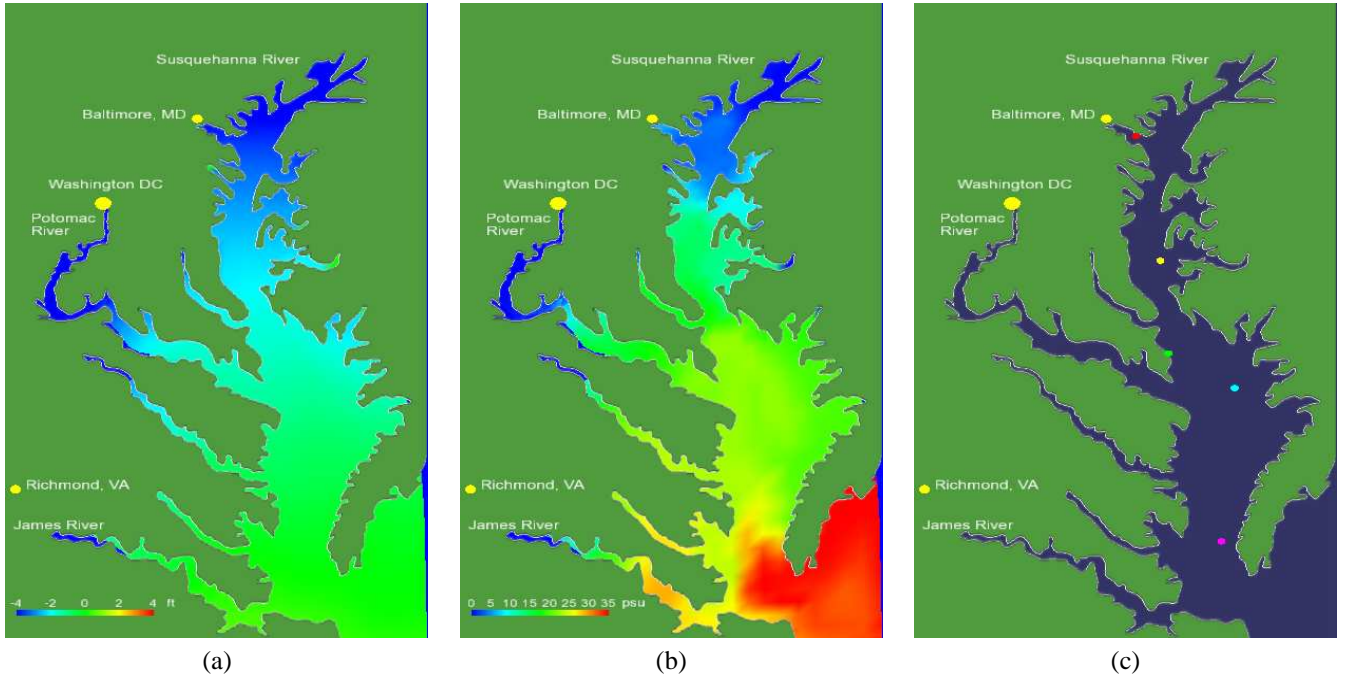


Figure 3. Visualization of (a) Water height, (b) Salinity, and (c) Floating particles.

## 2.2 Visualization Design and Implementation

The computation results returned by iROMS to the visualization module are only defined at the grid points, which have the dimensions  $20 \times 50 \times 10$  for the Chesapeake Bay. Direct visualization of salinity, water surface height, or other scalar quantities as a triangular or quadrilateral mesh formed by the computation grid is not visually pleasing due to the small number of grid points. A direct visualization of salinity is shown in Fig. 2(b), where the bay is assumed to have uniform salinity. While this might be acceptable for a professional oceanographer, it is apparent that better visualization techniques are needed in order to make the program appealing to casual users and the general public.

A texture image was first created based on an accurate vector representation of the boundary of the Chesapeake Bay in the form of a digital line graph. The texture image is shown in Fig. 4(a), where the land around the Chesapeake Bay is shown in green and the rest of image is the bay itself. In addition to the red, green, and blue components, the texture image also has a fourth component, the alpha component. The alpha values corresponding to the land area are one (opaque), while the alpha values corresponding to the bay are zero (transparent). The texture image was then applied to a rectangle that covers the entire area of interest in the visualization, and displayed on the top of the triangular mesh formed by the computation grid. The resulting visualization is shown in Fig. 4(b). Close examination of Fig. 4(b) reveals that the visualization is incorrect around the boundary of the bay. An enlarged view of the area labeled by the white box in Fig. 4(b) is shown in Fig. 5(a).

The blue jaggedness in Fig. 4(b) and Fig. 5(a) was caused by the mismatch between the high-resolution bay boundary in the texture image and the coarse computation grid used by salinity or surface height visualization. In other words, the blue jaggedness is the aliasing effects due to the limited resolution of model grid. The bay's area in the computational grid is smaller than its area in the texture image.

To eliminate the aliasing effects in the visualization, the size of the Chesapeake Bay returned by iROMS is enlarged by changing land grid points at the edges of the bay to water grid point in the visualization module. A grid mask is used by both iROMS and the visualization module for differentiation between land and water, i.e., a water grid point has a mask of 1 while a land grid point has a mask of 0. A  $3 \times 3$  window is used to search for land boundary points. If any of the four direct neighbors (top, bottom, left, right) of a land grid point is a water grid point, then the center land grid point is a land boundary point. In Fig. 6, the water grid points are represented by blue dots, land grid points green dots, while the land boundary grid points green dots with blue circles around them.

After land boundary grid points are identified, the variables at those points are recomputed as the average values of their direct neighbors. Denote  $\rho$  as a physical variable at a land boundary grid point, its new value is computed as

$$\rho_{\text{boundary}} = \frac{\sum_{i \in \Omega} \rho(i) M(i)}{\sum_{i \in \Omega} M(i)} \quad (1)$$

where  $\Omega$  is the set of grid points that are directly connected to the center grid point and  $M(i)$  is the water mask at grid point  $i$ . The salinity and water height are processed us-

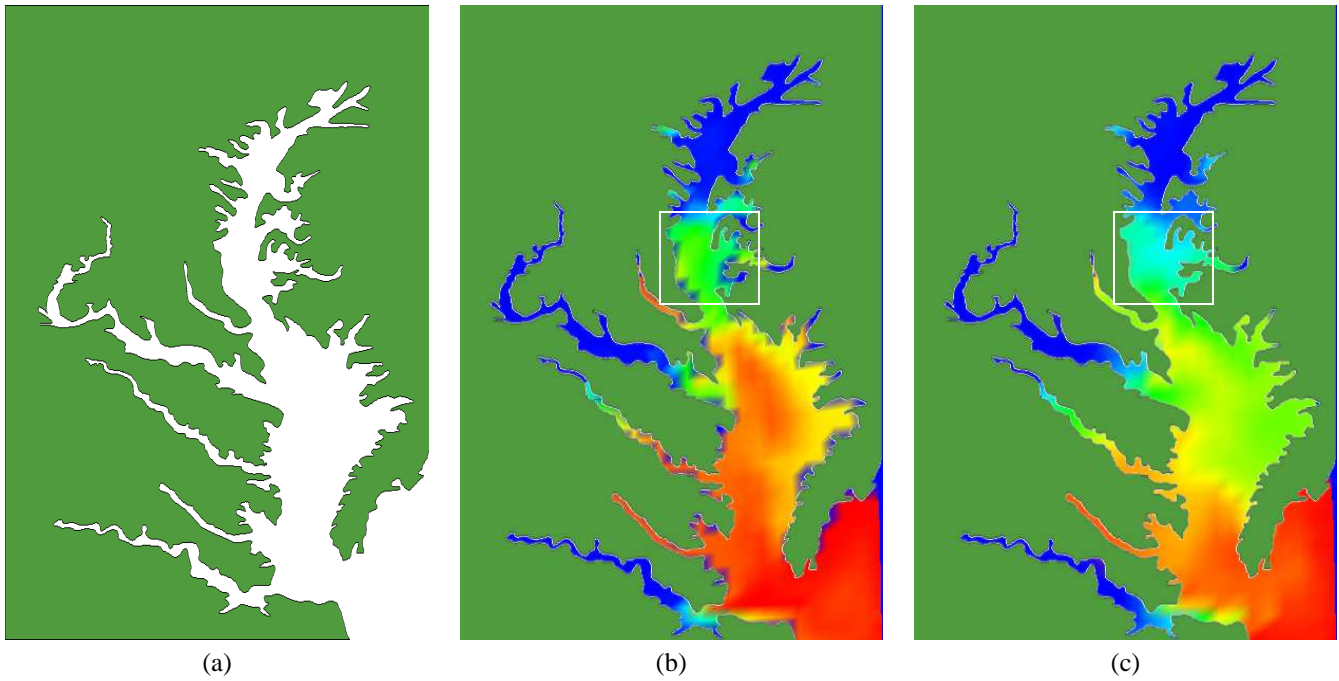


Figure 4. (a) Texture image containing a high-resolution boundary of the Chesapeake Bay. (b) Aliasing effects due to the coarse resolution of the computation grid. (c) Aliasing effects are removed by "enlarging" the bay (changing land boundary grid points to water grid points). Note that (b) and (c) were captured at different times, but the aliasing effects in (b) do not vary with time.

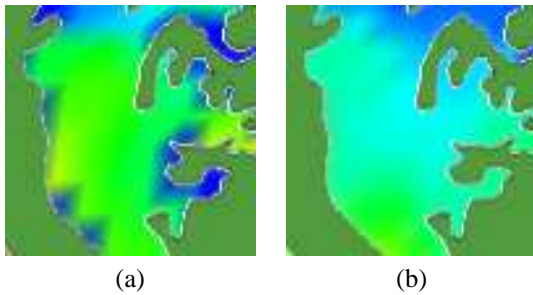


Figure 5. Enlarged views of (a) Fig. 4(b), and (b) Fig. 4(c).

ing equation (1). The improved visualization results are shown in Fig. 4(c), and Fig. 5(b) shows the enlarged view Fig. 4(c). The figures demonstrate that the previous jaggedness is removed by enlarging the bay and averaging neighboring grid points.

Annotations for the major rivers flowing into the Chesapeake Bay as well as several important places were created to facilitate user orientation. Color map scales were included for salinity, height, and velocity. To generate a correct final image, all objects must be rendered in a proper order. The correct rendering order is illustrated in Fig. 7, with the bottom layer rendered first. Thus, the enlarged triangular mesh formed from the computation grid is rendered first, then the rectangle containing the bay boundary texture image, and finally the annotations and colormap scales. Different layers are put into different renderbins provided

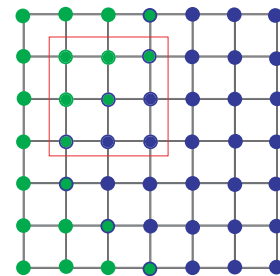


Figure 6. Searching for boundary grid points. The bay is represented as blue dots and the land is represented by green dots. The land boundary is represented by green dots with blue peripheral.

by OpenSceneGraph to guarantee the correct rendering order.

### 3 Simulation and Visualization Results

The integrated simulation and visualization environment allows users to interact with an ocean circulation model in a more cohesive way than previous approaches with separate simulation and visualization. Running iROMS as the numerical kernel for ocean modeling, the interactive visualization program is capable of demonstrating many interesting ocean phenomena. Here we briefly describe several typical phenomena that exist in the Chesapeake Bay.

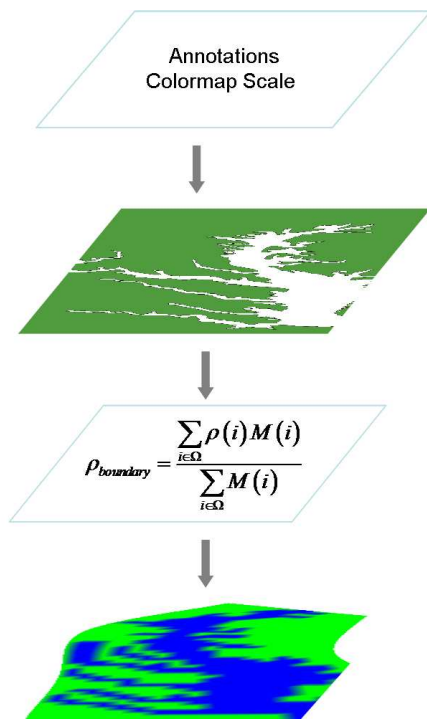
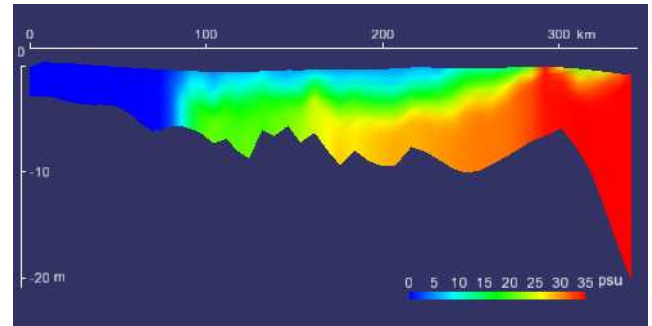


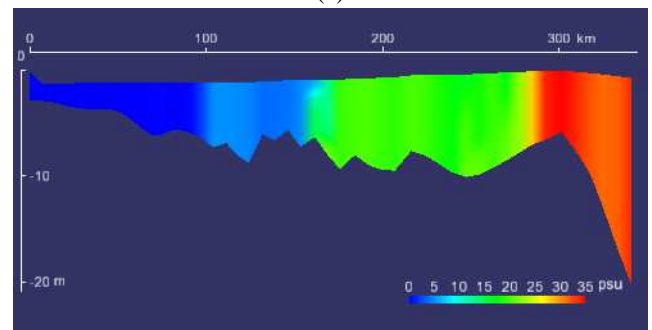
Figure 7. Rendering order with the bottom layer rendered first.

Estuaries such as the Chesapeake Bay are continuously fed by a strong source of fresh water at their head. The normal estuarine circulation consists of upper layers of relatively fresh water flowing out of the bay and lower layers of saltier water flowing up into the bay, due to the fact that saline water has a higher density than fresher water. This salinity stratification is apparent in Fig. 8(a), which also shows a boundary between fresh water and salty water near the head of Susquehanna River that fluctuates with the tides. The plan view shown in Fig. 9(a) shows low salinity at the head of the estuary and high salinity at the mouth, where oceanic water is being pulled into the estuary. Fig. 9(b) shows the velocity of the top layer corresponding to the fresh water flowing out of the bay, while Fig. 9(c) shows the velocity of the bottom layer corresponding to the saltier water flowing into the bay.

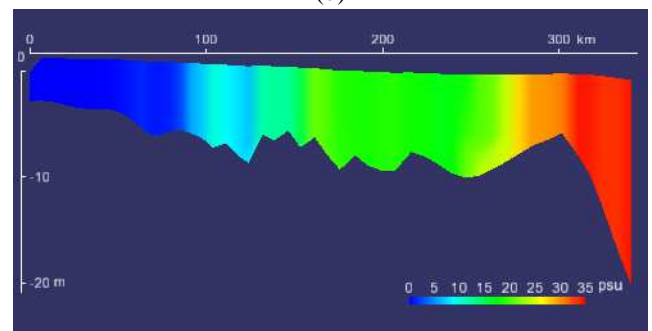
During the summer, the Chesapeake Bay has few strong wind events, and the stratification in the central bay becomes quite strong. This isolates the bottom layer away from the surface, so that it is not in contact with the atmosphere. Combine this with bacterial decomposition of detritus, and the oxygen can be depleted in the lower layer of water, making the bay uninhabitable for fish and other animals. A severe storm can mix the water column and provide oxygen to the whole bay. The stronger the wind, the quicker this mixing takes place. Fig. 8(b) shows the mixing of water columns caused by strong wind from the north. It can be seen that the water surface height is depressed by the wind from north. In Fig. 8(c), the surface



(a)



(b)



(c)

Figure 8. Salinity stratification and wind effects. (a) Salinity stratification is the normal state of estuarine circulation of the Chesapeake Bay. (b) Strong wind from north breaks the salinity stratification by mixing water columns. (c) Strong wind from south can cause flooding by raising water surface height.

height near the head of the estuary is significantly raised by a strong wind from south, giving rise to severe flooding.

Tides are easily observable in the interactive simulation and visualization program. Tides in the open ocean are formed due to the gravitational influence of the moon and the sun. Most of the tidal rise and fall in the Bay is driven by the tidal behavior of the water in the open ocean at the mouth of the Bay. Tides are most apparent in the depth view, where the the surface height exhibits periodic variations. Tides give rise to currents which flood into the bay before high tide and ebb out of the estuary after high tide. The particle tracking option (Floats) in the plan view gives users a clear view of the particle (water) movements due to the tides. Tides are important for determining the level of mixing in an estuary. There are many other interesting phenomena that can be demonstrated by the interactive simulation and visualization program, such as plume structure, upwelling, and downwelling. In addition, the program has several built-in scenarios that users can run directly, including normal spring, normal summer, rainy spring, etc.

## 4 Conclusion

This paper discussed interactive visualization of Regional Ocean Modeling System applied to the Chesapeake Bay. Portions of ROMS were modified to facilitate user interaction. Users can control the forcing conditions applied to ROMS through a graphical user interface and observe and examine the results in real-time. Texture mapping, alpha-channel blending, and filtering techniques were used to create a smooth boundary for the bay and eliminate the aliasing effects due to a low-resolution grid. The integrated simulation and visualization environment allows a user to control wind speed and direction along with the rate of flow from the rivers that feed the bay and provides visualizations of water height, velocity, salinity across horizontal and vertical planes, and particle tracking.

The current integrated simulation and visualization environment can be enhanced in several ways. First, it can be extended and applied to other bodies of water modeled by ROMS. Instead of building a new program for each location, a configuration file can be used to encapsulate all the pertinent information about a model's geometry, grid characteristics, and forcing conditions, and different user interface and visualizations can be generated for different locations according to their specification. The second enhancement is expansion of the set of ROMS features that are accessible through the interactive environment. Visualization of pollution and oxygenation distributions can be easily added, along with user controls to govern the number, location, and type of pollution sources and the computational grid dimensions. Other visualization methods can be investigated for possible improvements, such as point-based rendering, GPU-based programmable shaders.

## 5 Acknowledgements

This project was sponsored by the Virginia Governor's Research Initiative through the Office of Research at Old Dominion University. The authors thank John Klinck, Lee Belfore, and Elizabeth Smith for their assistance.

## References

- [1] K. Bryan, "A numerical method for the study of the circulation of the world ocean," *Journal of Computational Physics*, vol. 4, no. 3, pp. 347–376, 1969.
- [2] A. J. Wallcraft and D. R. Moore, "The NRL layered ocean model," *Parallel Computing*, vol. 23, pp. 2227–2242, 1997.
- [3] A. F. Blumberg and G. L. Mellor, *Three-Dimensional Coastal Ocean Models*. Washington, DC: American Geophysical Union, 1987, ch. A Description of a Three-Dimensional Coastal Ocean Circulation Model, pp. 1–16.
- [4] A. F. Shchepetkin and J. C. McWilliams, "The regional oceanic modeling system (ROMS): A split-explicit, free-surface, topography-following-coordinate oceanic model," *Ocean Modelling*, no. 9, pp. 347–404, 2005.
- [5] J. C. Warner, C. R. Sherwood, H. G. Arango, and R. P. Signell, "Performance of four turbulence closure methods implemented using a generic length scale method," *Ocean Modelling*, no. 8, pp. 81–113, 2005.
- [6] A. Johannsen and R. J. Moorhead, "AGP: Ocean model flow visualization," *IEEE Computer Graphics and Applications*, vol. 15, no. 4, pp. 28–33, July 1995.
- [7] S. Nations, R. J. Moorhead, K. Gaither, S. Aukstakalnis, R. Vickery, W. C. Couvillion, D. N. Fox, P. Flynn, A. Wallcraft, and O. M. Smedstad, "Interactive visualization of ocean circulation models," in *Seventh IEEE Visualization*, 1996, pp. 429–432.
- [8] G. M. Nielson, H. Hagen, and H. Muller, *Scientific Visualization: Overviews, Methodologies, and Techniques*. Los Alamitos, CA: Institute of Electrical and Electronics Engineers, March 1997.
- [9] H.-W. Shen, G.-S. Li, and U. D. Bordoloi, "Interactive visualization of three-dimensional vector fields with flexible appearance control," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 4, pp. 434–445, July/August 2004.
- [10] Z. Liu, R. J. Moorhead, and S. B. Ziegeler, "Ocean flow visualization in virtual environment," Mississippi State University, Technical Report, 2003.

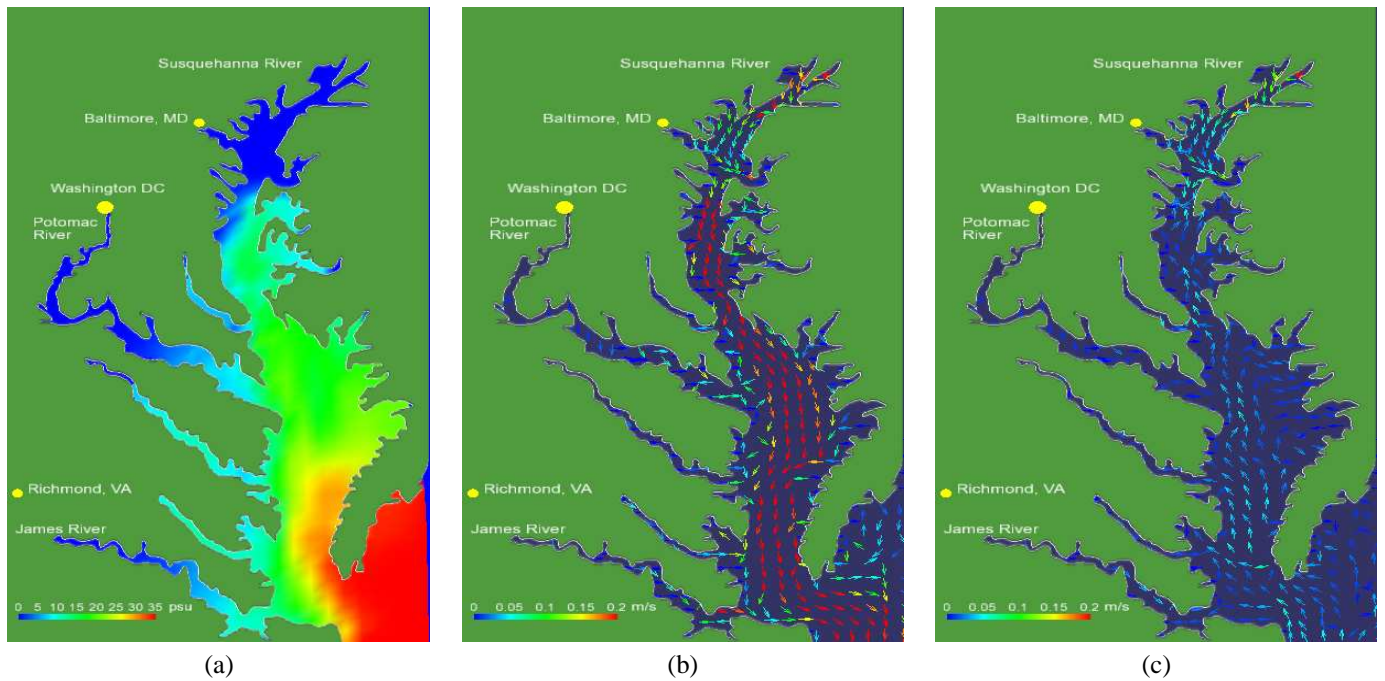


Figure 9. The normal state of estuarine circulation of Chesapeake Bay. (a) The head of the estuary has low salinity while the mouth of the estuary has high salinity. (b) The top layer contains fresh water that flows out of the bay. (c) The bottom layer contains saltier water that flows into the bay.

- [11] K. Gaither, R. J. Moorhead, S. Nations, and D. Fox, "Visualizing ocean circulation models through virtual environments," *IEEE Computer Graphics and Applications*, vol. 17, no. 1, pp. 16–19, January-February 1997.
- [12] S. A. Brown, M. Folk, G. Goucher, and R. Rew, "Software for portable scientific data management," *Computers in Physics*, vol. 7, no. 3, pp. 304–308, May/June 1993.
- [13] M. Snir and W. Gropp, *MPI: The Complete Reference*. The MIT Press, 1998.
- [14] W. G. Large, J. C. Williams, and S. C. Doney, "Oceanic vertical mixing: a review and model with a nonlocal boundary layer parameterization," *Reviews of Geophysics*, vol. 32, no. 4, pp. 363–403, 1994.
- [15] FLTK, <http://www.fltk.org>.
- [16] OpenSceneGraph, <http://www.openscenegraph.org>.