04JEI00026
1726-2135/1684-8799

www.iseis.org/jei

Journal of Environmental Informatics 3 (1) 42-50 (2004)

An Integrated Approach to Real-time Environmental Simulation and Visulization

B. Huang[1*], D. Xiong[2] and H. Li[3]

[1]Department of Geomatics Engineering, University of Calgary, AB T2N 1N4, Canada
[2]Engineering Science and Technology Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[3]Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing 100101, China

ABSTRACT. This paper introduces a novel visualization approach that can effectively facilitate the analysis, control, and refinement of dynamic environmental simulations on the World Wide Web. This approach overcomes drawbacks of current Internet Geographic Information System (GIS) technologies by providing an effective and efficient mechanism for two-way and sustained communication and synchronization between the visualization and the modeling processes. The critical aspect of this approach is the establishment of a virtual environment on the Internet using the applet-servlet-socket architecture that supports real-time interactive and collaborative visualization of an environmental modeling process. In this virtual environment, the model residing in the simulation application server is fed with real-time rainfall data from a remote data server through a socket connection. The computational modeling and visualization can take place simultaneously on the application server and the client sides. As modeling computations proceed on the server, modeling results and outputs stream into the client side continuously. The client interface is updated with live 3D displays (not in the sense of predefined 3D animations provided by AVI or dynamic GIF files). Meanwhile, these 3D graphics can act as a support for further user interactions. A hydrological model, TOPMODEL, is implemented using the proposed web-based visualization environment to demonstrate the applicability of the proposed approach in facilitating environmental modeling and simulation.

Keywords: Environmental simulation, internet, applet-servlet-socket, TOPMODEL, visualization, VRML

1. Introduction

People visualize by nature, and visualization has become an integral part of scientific data modeling and analysis (Marshall et al., 1990). In environmental planning and management, using visualization tools as an aid to understand complex environmental data has received a great deal of attention (Rhyne et al., 1993). While static visualizations of environmental phenomena are still valuable for the objectives of many studies, there is an increasing demand for dynamic, interactive visualization capabilities to facilitate the analysis and understanding of complex environmental processes (Burrough, 1998). In particular, there is a strong interest in using advanced visualization techniques to involve public interests in environmental decision making as visualizations provide a powerful means through which environmental scientists and the general public can effectively communicate and collaborate.

Continuing advances in the technologies of computer networking and interactive graphics (Singhal & Zyda, 1999; Walsh & Bourges-Sevenier, 2001) make the dynamic, visual exploration of environmental processes particularly attractive. Interactive control and visualization, or process steering which links visualization with computation, has been employed to explore the virtual world created from environmental models in a single user environment (Marshall et al., 1990). However, making such an interactive visualization to a larger group of end users is still an open challenge. The emergence of the Internet as a fast and efficient information medium offers some exciting new opportunities to address this challenge. On the one hand, interactive simulation and visualization on the Internet provide simultaneous access to many users, and this access is on-line and in real-time. The simulation and the visualization can be carried out either through a live broadcast or through the interactive control by the end user. On the other hand, not only environmental scientists can take advantages of this capability to effectively communicate their data, results, and understandings among themselves, but the public also can use the Internet to appreciate and evaluate the same information. While it is an appealing idea to make environmental models operational across the Internet, the implementation of such a capability is a non-trivial task. Major difficulties for Internet-based combined environmental modeling and visualization include:

- Given the computational intensity required for environmental modeling and visualization in general, computational resources must be carefully allocated in order to achieve the optimal use of these resources. It is still a constant struggle to determine whether an application (e.g., a modeling procedure or the visual representation

* Corresponding author: cvehb@nus.edu.sg

of the modeling results) should be run on the client side or on the server side.

- Environmental modeling and visualization involve massive amounts of data. If not effectively managed, inefficient data communications between the server and the client can easily jam the network. Proper mechanisms must be devised to effectively control and schedule bi-directional communications and data exchanges during the modeling and visualization process. For instance, when modeling results are derived from the server side, these results must be packaged and delivered in a specified time interval to minimize data volumes. At the same time, information contained the data must be retained, and a proper feeding rate to the client must be maintained to make the process appear "alive".
- Dynamic simulation and visualization require frequent graphical updates on the screen of the client side. These updates, if not correctly handled, would involve excessive amounts of redundant data. To overcome this problem, graphical updates must be implemented incrementally so that existing displays (e.g., a 3D visualization and a chart) are only updated in places where changes occur at each time interval. To do so, only part of the graphical data is reloaded and redrawn.
- Synchronization among different tasks while simulation and visualization occur concurrently across the network is yet another challenge. To address this challenge, system design and implementation must establish effecttive ways for message exchange and task coordination among various activities (e.g., user inputs, model calculation, and graphical display).

Because of these difficulties, much visualization developed on the Internet is often limited to one-directional communication between the server and the client. In this one-directional communication, the server will respond to a request from the client by providing a static scene to the client. As such visualization is not truly interactive, end users will be limited to accessing a set of predefined outputs. It is most desirable, however, that the end users can actively interact with the simulation process, control and steer this process so that they will be able to explore different scenarios and derive information to meet their own requirements. Nevertheless, the implementation of high-level of user interactions and visualization steering requires frequent two-way communications between the client and the server. Existing Internet-based visualization technologies such as Common Gateway Interface (CGI) or Active Serve Page (ASP), in combination with Virtual Reality Modeling Language (VRML), have shown various limitations in dealing with process-based visualization issues (Huang et al., 2001).

This paper introduces a real-time visualization approach to environmental modeling and simulation. This approach facilitates the establishment of a virtual environment on the Internet that supports interactive and collaborative visualizations of dynamic environmental modeling processes. In this virtual environment, the (application) server is supplied with raw data in real-time. The computational modeling and visualization can take place simultaneously on the server and on the client sides. As modeling computations proceed on the server, modeling results and outputs stream into the client side continuously. The client interface is updated with live 3D displays (not in the sense of predefined 3D animations provided by AVI or dynamic GIF files). Meanwhile, these 3D graphics can act as a support for further user interactions.

The research takes a hydrological process model as an example to demonstrate the implementation and effectiveness of the proposed approach. This demonstration is based upon the use of the TOPMODEL (TOPography based hydrological MODEL) (Beven & Kirby, 1979). TOPMODEL is a well-known hydrological model that is able to make computationally efficient predictions of stream flows, which simulates catchment response to rainfall input. Integrating the model with the visualization environment as proposed by the research, the predicted hydrological responses can be visualized using 3D maps on the Web and the modeling process can be effectively controlled by the end user.

The remainder of this paper is organized as follows. Section 2 presents the underlying visualization principles of our approach, and how it can be used for a better understanding of complex phenomena by implementing it in a highly interactive computer environment on the Internet. Section 3 introduces the communication and visualization techniques used in this research. In section 4, a brief overview of TOPMODEL is given. Section 5 presents an Internet implementation of the TOPMODEL hydrological model with frequent communications between the client and the server. Section 6 describes the performance test on the program developed. Finally section 7 draws some conclusions.

2. Interactive Simulation and Visualization

Data analysis and visualization play critical roles in the environmental modeling process. Traditionally, data analysis and visualization are performed as post-processing steps after simulations have been completed. As shown in Figure 1, the visualization part can be described well by a simple form of the visualization model in Upson (1989), i.e. the visualization pipeline model (see also Haber & McNabb, 1990). This model views visualization as a pipeline in which the simulation data is fed in, and successively filtered, mapped, and rendered to create a displayable image.

The filter process accesses the raw data or simulation data and operates on and modifies them in one or more ways to derive data for subsequent visualizations. For example, a soil saturation deficit matrix is selected from the simulation result of TOPMODEL. The map process creates an abstract geometrical representation of the data, e.g., a saturation isoline map, and the rendering process takes the 3D geometry from the map process and applies lighting, shading, and projection to create an image.

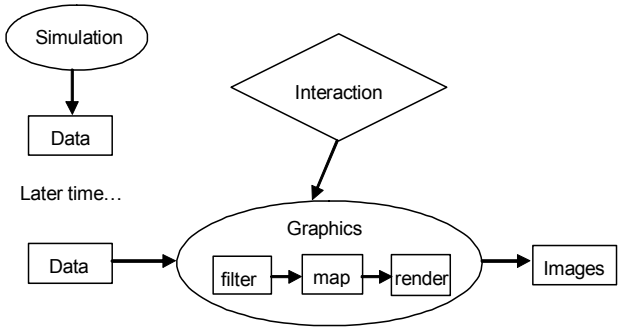

Figure 1. Post-processing.

The main limitation of this approach is that model errors can be found only during post-processing. Also the degree of user interaction is quite low.

Ideally a dynamic and interactive visualization process, as shown in Figure 2, is established so that the modeler can steer both the simulation and visualization processes as computations proceed. Such a steering technique allows modelers to monitor their environmental modeling programs while viewing the results of the calculation graphically, thereby facilitating the program debugging. On the other hand, it also allows users to fully control model parameters to observe the model behavior through visual presentations, and thus enhancing understanding of the modeling process.

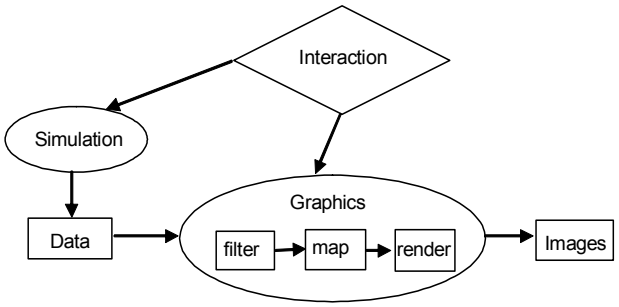

Figure 2. Interactive simulation and visualization.

Compared with the post-processing model, this model provides a mechanism through which the user is able to control and manipulate the proceeding of the simulation and may then alter the processing steps as required. This process is continuous, dynamic and interactive (Hibbard, 1998; Johnson, 1999).

The above model is also amenable to incorporating different web-based visualization options and strategies in terms of filtering, mapping, and rendering tasks.

3. Bi-directional and Real-time Communications and Web 3D-based Visualization

Traditionally, web-based visualizations are often performed on either the server side or the client side, which results in a fat server or a fat client accordingly (Huang et al., 1999). For environmental modeling and visualization, as frequent communication between computation and visualization is needed, we advocate a relatively balanced and synergic client/server application, in which the client is dedicated to visualizations, while the server is responsible for intensive modeling computations, and the client and the server can maintain a continuous communication until a modelling process finishes. The hybrid approach exploiting both the client and server sides has already been explored (e.g., Bender, 2000; Huang et al., 2001), which provides greater flexibility for task sharing between the client and the server and the enhanced capability for implementing visually attractive interfaces. While this approach alleviates the problems of the single side method, it is still not ideal, as it cannot achieve bi-directional and sustained communications between the client and the server. In this regard, we explore an architecture which supports bi-directional communication and 3D visualization over the Internet.

3.1. Bi-directional and Persistent Communication

Since the traditional approaches have their limitations for applications in environmental simulation and visualization, we become particularly interested in using a server technology, i.e., servlets, to achieve bidirectional communications through data streaming.

Servlets are not new. A servlet is a generic server extension, a Java class that can be loaded dynamically to extend the functionality of a web server. It can be thought of as a server side applet because a servlet can extend the capabilities of a server in the same way as an applet extends the capabilities of a browser. A Java servlet runs inside a Java virtual machine (JVM) on the server.

The applet connected with the Common Gateway Interface (CGI) program is not suitable for intensive communications between the server and the client, as new communication channels must be established for each request and response. The introduction of Java servlets and object serialization has given a second life to these traditional applet-server communications techniques. Servlets replace slow-starting CGI programs, which improves the performance of applet-server communication and make frequent applet-server communications feasible. All these lend the applet-servlet approach to dynamic modeling and visualization because efficient bi-directional communications are essential for dynamic modeling and visualization.

The applet and the servlet can interact in several ways, including HTTP text, HTTP object, socket, and RMI (Hunter & Crawford, 1998). In our case, we choose an HTTP object because it is simple, intuitive, and standard. Also, it can be immune from firewall blockage and browser security checks.

Obviously, other communication methods can be utilized as well. For example, a socket connection can be established to deliver data streams between the applet and the servlet that may allow more efficient bi-directional, sustained communications. The drawback however, is that decoding these data streams may be time-consuming. More seriously, the socket connection may fail for applets running behind firewalls, as most firewalls disallow raw socket connections.

While the socket connection is not used in the client-(application) server communication for complying with the generic HTTP protocol, it is used for the real-time communication between a data server that stores the data and the (simulation application) server because such a communication does not need to use the Internet but just the normal local area network.

3.2. Web 3D-based Visualization

After the applet-servlet-socket architecture has been introduced, another major choice to make is on the presentation of 3D graphics using Web3D (Walsh & Bourges-Sevenier, 2001).

Web3D describes any programming or descriptive language that can be used to deliver interactive 3D objects and worlds across the Internet. These include open languages such as VRML, Java3D, X3D and any proprietary languages that have been developed for the same purpose. VRML and Java3D are widely used in the industry while X3D is the latest generation Web 3D technology that possesses great potential and flexibility.

Until recently, VRML has been the main standard for representing 3D content on the Web. It provides a versatile platform for a variety of applications that uses 3D as a central metaphor or interface. Major strengths of VRML include its tight integration with a variety of other Web technologies and its ease in incorporating the benefits of those technologies (e.g., its flexibility with various graphic, audio, and video formats; capability with scripting languages; and compatibility with different network protocols). Another powerful feature of VRML is its easy extensibility that allows new node types and capabilities to be added to the base language.

The Web technologies evolve quickly nowadays, and XML enjoys increasing popularity. XML and its extensions that support the description of geo-spatial data (e.g., GML) and query languages have been employed to describe various types of data in a number of application domains. The virtual reality (VR) community has, in fact, recognized the growing success of XML. In response, the Web3D Consortium, in concert with the W3C (World Wide Web Consortium), has defined an XML-compliant 3D standard for the Web, X3D ("Extensible 3D"). X3D extends the capabilities of VRML and provides a means to express the geometry and behavior of VRML using XML. It is now possible to translate a VRML file into a X3D file and each VRML node has a corresponding XML entity.

Nevertheless, the essence of VRML still remains. The old VRML syntax does not go away though it is encapsulated by XML. A VRML-based virtual environment can, therefore, be easily transplanted to the new X3D environment once the proposed standard is approved. In addition, VRML itself is continually evolving and is still the core of Web3D. Also, the VRML plug-ins are still widely used at present for 3D visualization on the Web. With all of these considerations, VRML is chosen for the presentation of 3D graphics in the current study. Java 3D is not selected because the software built upon it is relatively slow.

There are two different ways to bring dynamic contents into the VRML world (Figure 3). One is to implement JavaScript in the Script node, and the other is to control the VRML contents in a Java program using the External Authoring Interface (EAI). JavaScript, as included in a node of the VRML file, allows a good portability over the Internet. By using the Script node, the VRML scene is acting as the front-end that invokes user interaction and conveys 3D spatial information to background processes running in Java. However, there are also good reasons for the ability to control the scene content from outside the VRML browser, typically from within a web browser because this is an environment more users are familiar with. With this ability, a very flexible user interface can be created through an integration of VRML within a wider multimedia design and extended functionality, which provide capabilities for interactions and communications well beyond what the basic scene navigation controls can facilitate with VRML browsers (Brutzman, 1998). The EAI connects an applet with a VRML plug-in on the same page using browser-specific plug-in activation framework. Essentially, the applet has access to any nodes in the VRML plug-in that has been defined with the keyword “DEF”. Java from the applet can control any such nodes by sending them events that match event field types or reading any of the event fields’ or event Outs’ values. Since environmental simulations involve intensive computations of environmental models, and graphical displays, both inside VRML (e.g., 3D scenes) and outside VRML (e.g., charts), are required, a powerful user-control interface, utilizing the EAI approach to link and control these components, will provide a suitable solution.

As a result, the integrated VRML applet-servlet-socket architecture as shown in Figure 4 is adopted. In this architecture, the real-time data is fed into the (simulation) application server through a socket connection. The main computation of environmental modeling is undertaken by the application server, while 3D graphics rendering and updating, as well as synchronous updating of other graphic windows (e.g., a chart window) are performed by the client. The applet and the servlet communicate for a given time interval through an HTTP object, which is not of a large size.

Take the rainfall-runoff modeling and visualization as an example. Assuming that the data server is able to receive the rainfall data in real-time, and it continuously pushes the data to the application server. This server performs model computations such as calculations of soil saturation and flow generation, while on the client side, once the Java applet receives the computation results from the server, it will interact with the VRML scene of the watershed through the EAI and

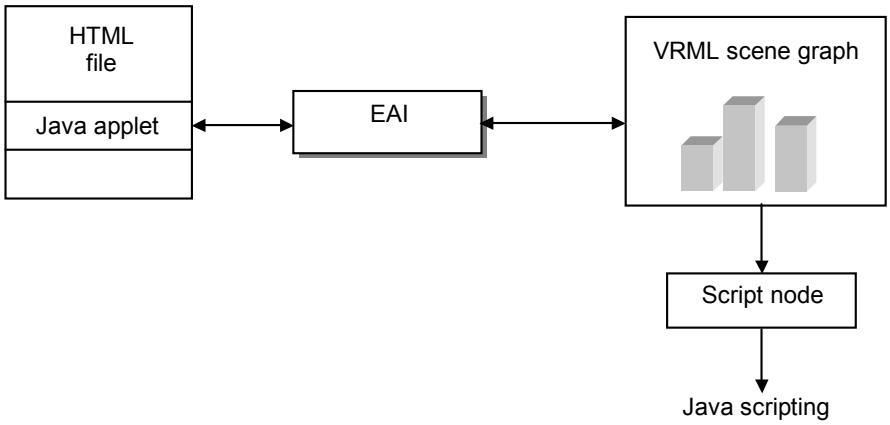

Figure 3. Java/VRML interaction.

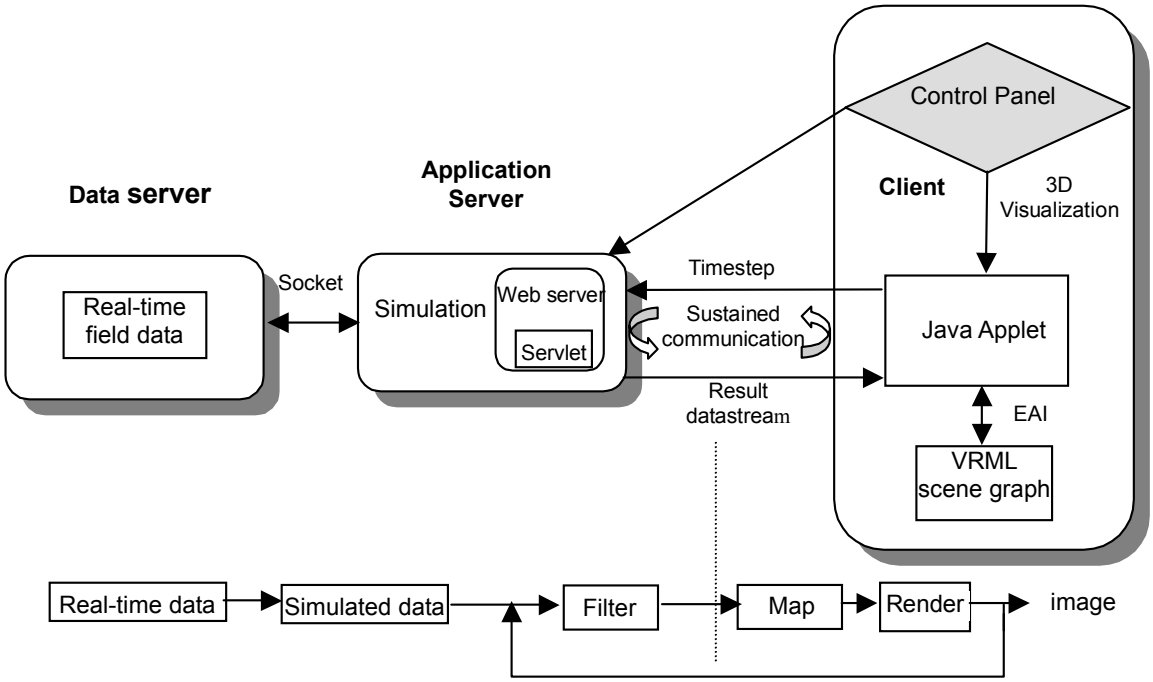

Figure 4. The applet-servlet-socket architecture for integrated simulation and visualization.

perform various operations on the scene, e.g., update the surface color. When this finishes, a signal will be sent to the server side, which then proceeds to the calculation in the next step. Such a process continues till the last step in the model. It is noted here that the real-time rainfall data pushing speed from the data server to the application server is generally faster than the applet-servlet communication. Thus, the latter does not need to worry about the real-time data pushing.

4. The Rainfall-Runoff Model: TOPMODEL

TOPMODEL (Beven & Kirby, 1979) is a physically based watershed model that simulates the variable-source-area concept of stream flow generation. This model requires Digital Elevation Model (DEM) data and a sequence of rainfall and potential evapotranspiration data. It predicts the pattern of soil water deficit S_i, as well as the resulting stream discharges. TOPMODEL has become increasingly popular as it provides computationally efficient prediction of distributed hydrological responses with a relatively simple framework using DEM.

This model is built upon the following equations:

$$\frac{1}{m}(\overline{S}-S_i)=(X_i-\overline{X})+(\log\overline{T}-\log T_i) \qquad (1)$$

A B C

and

$$X_i=\log(\frac{\alpha_i}{\tan\beta_i}),\overline{X}=\frac{1}{A}\int_A X_i,\overline{S}=\frac{1}{A}\int_A S_i,\overline{T}=\exp\left[\frac{1}{A}\int_A \log(T_i)\right]$$

where α_i is the local flow accumulation quantity, β_i is the local slope angle ($\tan\beta_i$ is an approximation of the local hydraulic gradient), T_i is the local soil transmissivity, $\overline{T}$ is the mean catchment transmissivity, $\overline{S}$ is the mean catchment soil water deficit, X_i is the local value of the topographic index, $\overline{X}$ is the mean catchment value of the topographic index, and m is a parameter dependent on the rate of change in hydraulic conductivity with depth. Equation (1) consists of three parts: a soil water-deficit distribution function, A, a topographic distribution function, B, and a soil-distribution function, C. If the spatial variation in soil properties is ignored, then the soil water-deficit distribution can be expressed as a function of the topographic index. The negative value of S_i indicates that the area is saturated, and the saturated overland flow is generated while the positive value of S_i indicates the area is unsaturated.

Over the past two decades, TOPMODEL concepts have been implemented with several computer languages like Fortran and Basic on different computer platforms (see e.g., Beven, 1997). The developed tools have been widely used in the application of hydrological modeling in numerous catchments in the world.

While TOPMODEL has been implemented in different ways, its prototype on the Internet has rarely been done so far. With the web-based modeling and visualization approach proposed in the previous section, the current research aims to demonstrate a different way for interactive and dynamic environmental modeling and visualization on the Internet.

5. Implementation of On-line TOPMODEL Simulation

The implementation of TOPMODEL mainly consists of two stages. The first stage is to derive α and tanβ from the elevation map of the watershed, which results in a topographic index map. The second stage integrates all hydrological parameters and rainfall records to predicate stream flows. The topographic index map is first derived by ArcView, a commercial-off-the-shelf (COTS) desktop GIS software package. The topographic index map is then read by the Java applet, which begins the simulation process.

Outputs of the TOPMODEL are designed to include different graphical displays, such as a hydrograph showing a curve of predicted stream flows or animation of soil saturation status at each time step. These outputs are intended for visual exploration of the modeling process and are generated by a Java applet at the client side.

To provide a realistic testing scenario, we use the datasets of the Slapton Wood catchment, Devon, UK. This catchment covers 0.94 km^2, 60% of which lies above the 90 m contour. The soils throughout the catchment are mainly freely draining acid brown soils with a clay-loam texture. These lend the catchment to the application of TOPMODEL.

The prototype interface is shown in Figure 5. The lower right window offers a control panel for users to interactively enter parameter values. After all the parameters are defined, the simulation process begins when the "simulation" button is clicked.

Firstly, a signal is sent to the real-time data server through the application server and the socket. Then the real-time data server starts to push the data to the application server till the data at all modeling steps are delivered.

Meanwhile, the Java-based client displays the 3D VRML model of the catchment draped by the topographic index map in the web browser (the left window). Then it sends the time step number to the servlet, which, after receiving the time step number, generates the saturated contributing area and the predicted stream flow at that time step with the data supplied from the real-time data server. The calculated result, encoded as an HTTP object, is then sent back to the Java client, which updates the 3D VRML model with the new saturated contributing area in the red color and draws the curve of predicted stream flows at the upper right window. This is achieved by updating a *color* node in the VRML model by a set of new color values as follows:

colorIn.setValue(update_color);

colorIn represents the color surface of the watershed. Upon completion of the color update (i.e. saturated cells are highlighted; the stream flow curve will also be updated). As shown in the upper-right part of Figure 5, the predicated stream flows move one step forward. As the background information of observed stream flows, and rainfall records are provided, the

difference between the predicted flow and the observed data can be easily compared. The displays of the soil saturation status and the predicated stream flows are synchronized using threads in Java programming. Such a process continues until the final modeling step is reached. It is worth noting that for the map generated on the client screen, only the portion that has been changed between two time steps will be updated. For instance, in the case of the saturated contributing area display, not the entire image will be sent from the server to the client for each time step. Instead only incremental changes on part will be sent and used as updates to refresh the client screen. This practice significantly reduces network traffic and increases computational performance.

With a proper screen-refreshing rate, continuous updates of the saturation map create an animation of the modeling process. The animation, in conjunction with dynamic stream flow drawing, provides a vivid graphical depiction of the hydrological process in the catchment. A user can also interactively adjust the parameters in the control window and view the corresponding changes of soil saturation status and predicted stream flows. With these capabilities, a user can easily investigate different hydrological responses while scenarios (e.g., rainfalls, terrain features, or soil conditions) are changed. Also the user might be able to fine-tune the modeling process so that the user's knowledge and judgement can be incorporated into the modeling results. For example, the value of *SRinit* (initial value of root zone deficit) is very sensitive to the prediction result since it determines the amount of saturated deficit that is essential to runoff generation in the TOPMODEL. If little changes take place on the value of *SRinit*, the observed difference between the observed and predicated stream flow curves can be much bigger. Through visual comparison of the differences between the observed and predicted stream flow curves, the user will be able to adjust the *SRinit* until the simulation results meet the user's satisfaction.

The visualization procedure even proves valuable for debugging the TOPMODEL. Through the comparison of predicated stream flow curves and the observed stream flows, we were able to identify and correct some of the program errors. Besides the ability for model parameter adjustment, a user can also control the simulation start and end steps through the timestep sliding bars and pause the simulation, if necessary. Furthermore, the user also has a degree of the control of the visualization process during a simulation. He/she can change visualization viewpoints, zoom in/out, pan, rotate, and tilt the 3D model. By doing so, the user will be able to view the modeling results in the way he or she prefers. More importantly, because of the use of the simulation and visualization approach as proposed by the current research, user interactions with the model simulation and with the visualization do not significantly impact the performance of the simulation and visualization process, which is critical in a web-based environment.

6. Performance Analysis

Performance testing has been undertaken on a PC installing Windows XP with Internet Explorer. The application server, the Jakarta server and the Tomcat servlet engine run on a Windows 2000 PC server with 1200 MHz CPU and 524 M RAM. The data server resides with a normal PC running Windows XP. The testing procedure was designed to mimic how the service would be used across the web. It differs from the procedure devised by Morrison and Purves (2002) in that we need to record the response time for a modeling process with a number of steps rather than one request-one result.

The procedure is that a client program submits a 'simulation' request to the application server, which computes streamflow and saturated cell numbers and then sends these to the client side. The client side updates the displays. Upon completion, the simulation proceeds to the next timestep. Such a process continues till the simulation finishes (there are in total 950 steps). The program was tested with different number of threads to simulate the running by single or multiple users. The client program can run in two ways—in single-thread mode, sending many requests in series, or in multi-thread mode, where many requests are sent in parallel. The second mode is more typical of the web, where many users may log on from many different machines, all at the same time. The testing program has been run multiple times and the response times logged. A typical distribution of response times is shown in Figure 6. The x-axis represents the response time for the client and the application server to finish 10 steps simulation and visualization, while the y-axis represents the percentage of performed steps (i.e. the steps that have results returned to the client) at a specific response time.

It is noted that we do not intend to make the visualization process very fast as we need to make sure that the display update at each modeling step be observed clearly by the user. Under this condition, the most rapid response times occur, as we would expect, are when the server is responding to a single thread (analogous to a single user accessing the service) where it achieves an average response time of 3.7s/step. As the number of threads increases (equivalent to concurrent users), there is some degradation in performance due to more consumption of computing resources. With 8 threads, the average response time is 7.5s/step.

7. Conclusions

This paper introduces an Internet-based approach for real-time environmental modeling and simulation using a VRML 3D-based applet-servlet-socket technique. This approach overcomes drawbacks of current Internet GIS technologies by providing an effective and efficient mechanism for two-way and sustained communications and synchronization between the visualization and the modeling processes. The interactions through simulation steering and visualization control enables users to experiment with alternative model computations, fine-tune model results, and visually observe the model behavior through dynamic 3D graphics. The critical aspects of the current approach are its ability to provide computational and communicational performances that are required for dynamic simulation and visualization in a

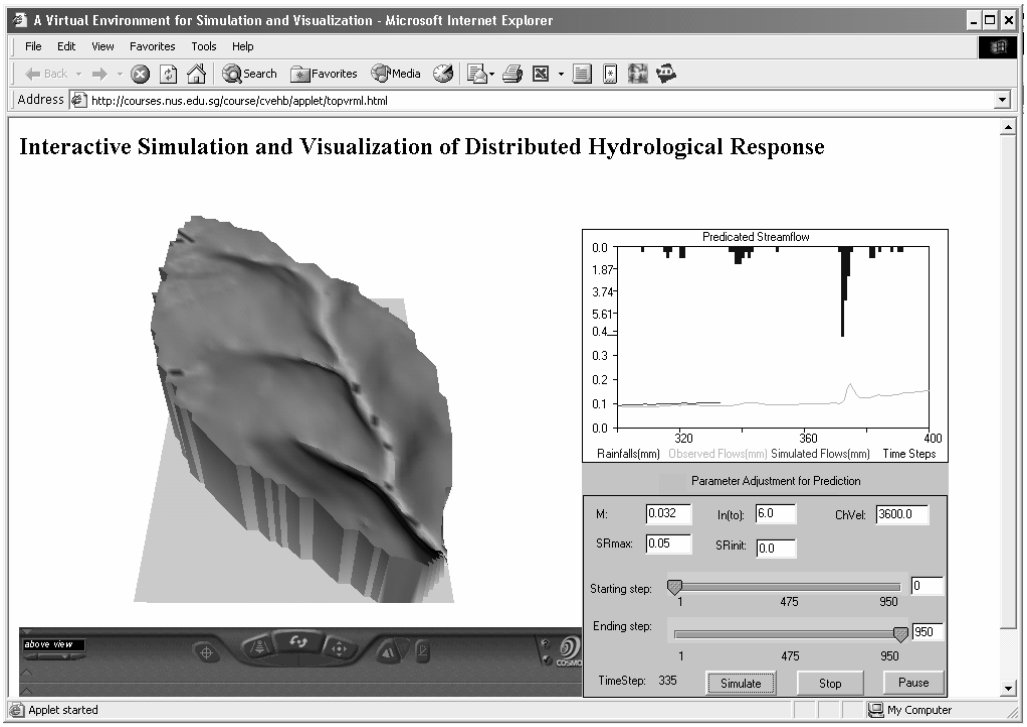

Figure 5. Steered simulation and 3D visualization.

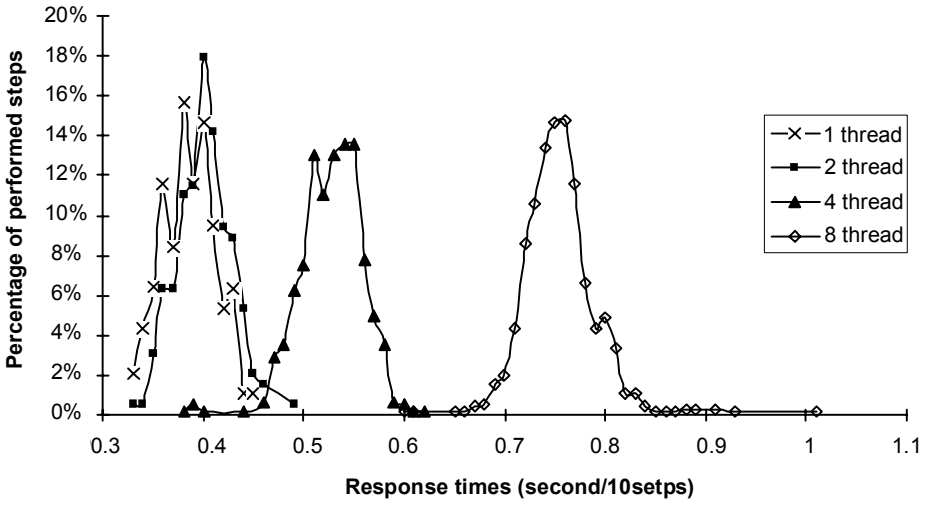

Figure 6. Distribution of server response times under different user modes.

web-based environment, and the ability to take in real-time data.

The implementation of TOPMODEL on the Internet has demonstrated the feasibility of the proposed modeling and visualization approach. It indicates that with the current approach, advanced environmental simulations and visualizations can be effectively carried out in a web-based environment. The ability to simultaneously simulate and visualize environmental processes on the Internet is particularly appealing because it can reach an audience that cannot be reached with other means. At the same time, the proposed approach provides similar capabilities that are available in a Desktop environment. Users are able to effectively steer the simulation calculations and exercise controls of the visualization process.

It has been frequently demonstrated that dynamic simulation and visualization can provide an elegant and powerful solution to the understanding of the environmental processes for both the environmental scientists and the public. From the perspective of environmental scientists, it helps to assess the evaluation of the convergence or divergence between simulated and observed processes in order to identify issues that are to be studied in future investigations or to formulate strategies in order to improve model procedures. From the perspective of the public, it helps them understand many of the environmental problems, encourage their participation in the environmental decision making process, and improve their ability to make the right decisions on many of the challenging environmental issues. The current approach contributes to environmental modeling and visualization in the sense that it allows broader collaborations and communications among environmental scientists and between environmental scientists and the public.

In conclusion, the implementation of the TOPMODEL on the Internet illustrates that the approach developed in the current research is effective and useful. It is hoped that the implementation techniques described in this paper can provide a basis for the implementation of other interactive and dynamic processes.

References

Bender, M., Klein, R., Disch, A. and Ebert, A. (2000). A functional framework for web-based information visualization systems. *IEEE Trans. Vis. Comput. Graphics*, 6(1), 8-23.

Beven, K.J. (1997). *TOPMODEL User Notes (Windows Version)*, Centre for Research on Environmental Systems and Statistics, Lancaster University, UK.

Beven, K.J. and Kirby, M.J. (1979). A physically based variable contributing area model of basin hydrology. *Hydrol. Sci. Bull.*, 24(1), 43-69.

Brutzman, D. (1998). Virtual reality modeling language & Java. *Commun. ACM*, 41(6), 57-64.

Burrough, P.A. (1998). Dynamic modeling and geocomputation, in P. Longley, S. Brooks, R. McDonnell and B. MacMillan (Eds.), *Geocomputation, a Primer*, John Wiley & Sons, pp. 165-191.

Haber, R.B. and McNabb, D.A. (1990). Visualization idioms: A conceptual model for scientific visualization systems, in B. Shriver, G.M. Nielson and L.J. Rosenblum (Eds.), *Visualization in Scientific Computing*, IEEE Computer Society Press, pp. 74-93.

Hibbard, W. (1998). VisAD: Connecting people to computations and people to people. *Comput. Graphics*, 32(3), 10-12.

Huang, B., Jiang, B. and Lin, H. (2001). An integration of GIS, virtual reality and the internet for spatial data exploration. *Int. J. Geogr. Inf. Sci.*, 15(5), 439-456.

Hunter, J. and Crawford, W. (1998). *Java Servlet Programming*, O'Reilly & Associates, Inc. Sebastopol, CA, USA.

Johnson, C., Parker, S., Hansen, C., Kindlmann, G. and Livnat, Y. (1999). Interactive simulation and visualization. *IEEE Comput.*, 32(12), 59-65.

Marshall, R., Kempf, J. and Dyer, S. (1990). Visualization methods and simulation steering for a 3D turbulence model of Lake Erie. *Comput. Graphics*, 24(2), 89-97.

Morrison, K.W. and Purves, R.S. (2002). Customizable landscape visualizations. Implementation, application and testing of a web-based tool. *Comput., Environ. Urban Syst.*, 26(2/3), 163-183.

Rhyne, T.M., Bolstad, M., Rheingans, P., Petterson, L. and Shackelford, W. (1993). Visualizing environmental data at the EPA. *IEEE Comput. Graphics Appl.*, 13(2), 34-38.

Singhal, S. and Zyda, M. (1999). *Networked Virtual Environments-Design and Implementation*, ACM Press.

Upson, C., Faulhaber, J., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R. and Dam, A. (1989). The application visualization system: A computational environment for scientific visualization. *IEEE Comput. Graphics Appl.*, 9(4), 30-42.

Walsh, A.E. and Bourges-Sevenier, M. (2001). *Core Web3D*, Prentice-Hall, NJ, USA.