An Architecture Supporting Live Updates and Dynamic Content in VRML Based Virtual Worlds *

Lee A. Belfore, II
Department of Electrical and Computer Engineering
Old Dominion University
Norfolk, VA 23529, U.S.A.
lbelfore@odu.edu

Keywords: VRML, live updates, dynamic content, servlets, GIS

Abstract

In this work, we present an application architecture that integrates geographic information systems (GIS) data into an application that supports several advanced capabilities. This application is implemented on the previously developed Interactive Land Use VRML Application (ILUVA). The application features the the integration of dynamic content generated by a server and support of live updates. From a user request, GIS layers are selected, translated from ESRI Shapefiles into VRML and then inserted into a world. The server architecture is designed for simple administration and maintenance using JavaTM servlets to field user requests and manage server data repositories. As an example, a world constructed to monitor a SuperFund cleanup site is presented.

INTRODUCTION

The rapid evolution of the Internet has provided opportunities for providing content in many different forms. The Virtual Reality Modeling Language (VRML) is an ISO standardized language that can be used to specify interactive virtual reality applications. Significantly, VRML has been designed to operate in the context of the Internet facilitating the delivery of virtual reality content over the Internet. In this work, we review an application architecture that integrates geographic information systems (GIS) data into an application that supports two powerful capabilities. First, a live update is the insertion of content into a world in such a way that the world does not need to be reloaded. Content that is inserted can be moved, modified, and can also interact with other aspects in the worlds. In our application, the new content is specified by a URL and can thus be retrieved from anywhere on the Internet. Second, dynamic content is the generation of content "on the fly" meaning that before a particular request for content, no file exists having the requested content. The content is synthesized on the server from foreign data sources and user configuration data. These two capabilities increase the flexibility and breadth of our application. Furthermore, we will give an overview of an application developed to monitor the progress of the Former Nansemond Ordnance Depot (FNOD) SuperFund cleanup site. The application features GIS registered imagery, synthesis of dynamic VRML content from ESRI Shapefiles, and the ability to create additional layers as live updates to the world. Other application capabilities include editing of the attribute data, selection of desired GIS layers, selection of desired imagery, and save/restore operations for operations that specify live updates.

Indeed, VRML has many desirable features that make it a compelling rendering technology for GIS applications [1, 2]. VRML offers basic capabilities of representing spatial data in three dimensions as well as functionality that allows implementation of observable changes to any geometry to display any associated attributes. The application domains are broad and include urban planning [3, 4] and geographic information systems (GIS) [1, 2]. Such applications require the manipulation and display of large data sets. The general approach followed is for the server to accept queries from client machine and then perform server side processing to satisfy the request. Each modification requires the entire world to be reloaded. For small data sets, reloading is not a significant issue; however, reloading large visualizations can add undesirable latencies. Several other researchers have developed notable VRML based GIS applications. In [2], primitives are defined that enable the creation of worlds described in true geocoordinates and the management of large data sets. Other applications provide mechanisms for managing the column of data that are typical of GIS applications.

Our VRML applications have evolved, first through development of the client part [3, 5] and then have been ex-

*This work was supported by the Army Corps of Engineers, Norfolk District, under USJFCOM Contract #N00140-97-D-2051/0018 and the Old Dominion University Virginia Modeling Analysis and Simulation Center (VMASC), Old Dominion University

tended to include client server applications [6, 7]. The client applications are stand-alone applications that support live updates, which is the ability to add features to the world without reload. The application integrates GIS and planning information for the theme of the application. To expand the capabilities, server processing has been developed to increase the flexibility and capabilities [5, 6]. The server processing provides the opportunity to enforce access control, selection of the content to be viewed, and generation of content dynamically. The application described in this paper, a visualization showing clean-up progress of a SuperFund site, extends prior efforts [3, 5–7] to create virtual worlds that allow configuration of a world and the translation of ESRI Shapefiles [8] into VRML. The virtual world was created to support public dissemination of information relating to cleanup efforts for an EPA SuperFund site over seen by the Norfolk Division of the Army Corps of Engineers. The Former Nansemond Ordnance Depot (FNOD) in Suffolk, Virginia was an ordnance site for World War I and World War II munitions [9].

This paper is divided up into five sections including an introduction, an overview of VRML, an outline of the software architecture, an example highlighting usage and a summary.

THE VIRTUAL REALITY MODELING LANGUAGE

The Virtual Reality Modeling Language (VRML) [10] provides a technological platform from which three dimensional models and worlds can be delivered over the Internet. VRML has the capability to define solid models and behaviors. For example, one can define the geometry of a solid model and also define how that model can change in response to user activity. The basic primitives in a VRML program are nodes. VRML is not optimized with any particular class of applications in mind. In this section, we describe some of the more relevant aspects of VRML as they relate to our application.

VRML has primitives, termed nodes, that describe solid models, sensors, viewpoints, lighting, interpolations, and arbitrary behaviors. Solid models are defined using the VRML `Shape` node in which the geometry and appearance of the model is defined. VRML offers a rich variety of geometry nodes from basic shapes (`Sphere`, `Box`, `Cone`) to complex and arbitrary (`IndexedLineSet`, `IndexedFaceSet`, `Extrusion`). In our application, we use `Sphere`, `IndexedLineSet`, `IndexedFaceSet`, and `Extrusion` nodes. The appearance of a shape can include simple coloration, indexed coloration, or textures (bit maps) painted on the shape surface. Sensors are used to sense the conditions at different locations within the world and also at different times. For example, drag and drop functionality can be implemented using a `TouchSensor` and a `PlaneSensor`. In addition, animations can be driven by a `TimeSensor`. Viewpoints are associated with certain strategic points within the world to aid the user in finding important information. Grouping nodes are used to define collections of nodes to be treated as one (`Group`, `Transform`) or special functionality for a collection of nodes (`Switch`, `LOD`). An interpolator will take a sampling of measurements and interpolate between samples. An animated viewpoint can be created using a `PositionInterpolator` holding a sampling of points along the desired path, a `TimeSensor`, and a `Translation` node. The nodes used in the animator are linked together using `ROUTE` declarations. Finally, with lighting nodes, the manner in which Shapes are illuminated are defined.

Arbitrary behaviors are treated separately due to their richness and complexity. `Script` nodes provide the opportunity to define general behaviors. A `Script` node consists of a list input, output and fields that are used within the script. The behavior is defined using either ECMAScript or Java, providing a great deal of flexibility in the behaviors. Several VRML specific methods are of value in the Script node. First, the `loadURL` method allows the script to load a URL that either replaces the world or loads in a named frame. In addition, the `loadURL` method can also be used to send messages to the server by appropriately assembling the information as parameters attached with the URL. Second the `createVrmlFromURL` method enables live updates because it makes possible the request and addition of VRML content to an existing world.

VRML includes several basic data types. These data types support the various data that necessary to define VRML nodes. Table 1 defines several data types. In addition to basic node types, VRML also provides the developer the ability to create user defined nodes. PROTO nodes are defined in the same file that they are instantiated, whereas EXTERNPROTO nodes are specified by a URL.

ARCHITECTURE

The application architecture is assembled to link three primary components. The first component is the domain data that is used to set the context for the world. The second component is the server that satisfies user requests, converts between data formats, and logs sessions. The third component is the client application that renders the world. The interrelationship of these three components is illustrated in Figure 1. At startup, the user requests the URL for the world. In response, the server provides both static information and dynamic content that is assembled in together to create the requested world. During a session, the user has the ability to make live updates, i.e. modify the world, save application information on the server, and restore a prior session.

Table 1. Selected VRML Data Types

Name	Type	Note
SFBool	Boolean	TRUE/FALSE
SFInt32	32 bit integer	in range $[-2^31, 2^31)$
MFInt32	Array of SFInt32	
SFFloat	32 bit floating point	Single precision (32 bit) IEEE
MFFloat	Array of SFFloat	
SFVec3f	3D Cartesian coordinate	Length 3 array of SFFloat
MFVec3f	Array of SFVec3f	
SFColor	Color Red, Green, Blue	Length 3 array of SFFloat constrained to [0,1]
MFColor	Array of SFColor	
SFRotation	Rotation in 3D space	Length 4 array of SFFloat first three must form a unit vector
MFRotation	Array of SFRotation	

Domain Data

The domain data encompasses the different data components that are necessary to create a credible visualization. Indeed, the domain data is what ultimately makes a particular deployment useful. Providing a mechanism by which this data is seamlessly and automatically integrated together broadens the utility of the architecture. In our application, this data consists of the GIS data to describe the target area. The data has been provided in the ESRI Shapefile format [8]. The Shapefile format is a combined database that includes information about the different feature shapes and the attributes associated with the shapes. Attribute queries can be made to select the matching shapes. In addition to the Shapefiles, geographically registered images provide realistic appearance when draped over a terrain. Clearly, the domain data is not in a format that is directly compatible with VRML and requires conversion. The conversion process will be accomplished on the server. Typically, the domain data is numerically represented as double precision values.

Server Architecture

A key challenge in creating visualizations is being able to provide timely, up to date information to clients automatically. In order to achieve this flexibility, the server needed to be able to determine what information was available on the server and then present the appropriate choices to the user upon logging in. Furthermore, the maintenance procedures for updating and adding information should be simple and reliable.

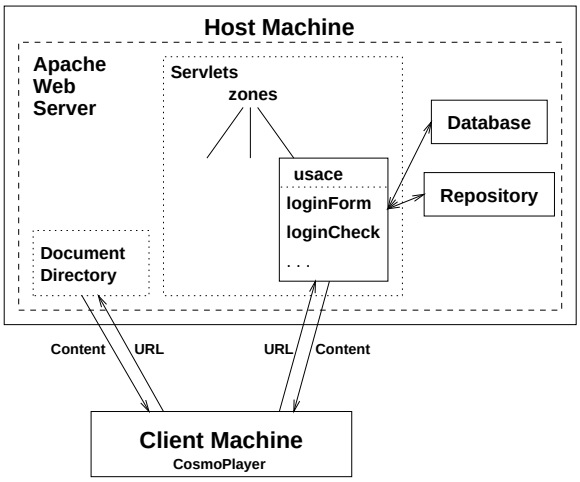


Figure 1. Top Level Architecture

Overview

Table 2 shows a high level script listing steps involved in a session. The client and server have specific activities that occur in each step. In addition, the administrative aspects are described in the third column. In addition, the administrative tasks are shown by each user task. The server architecture was built upon the Apache web server and ApacheJServ servlet engine. Apache provides all of the necessary capabilities to serve files on the Internet. ApacheJServ enables the coding of servlets, light weight Java methods, for access control, session control, generate dynamic content, and log user activity. The server architecture was designed to make update of current and development of alternate deployments as easy as possible. Our application builds on server functionality described in [5] and adds significant capabilities for integrating dynamic content.

Login: Managing Access Control

Users may log in providing them with the ability to select the desired information. Alternatively, the users may select a preconstructed world. In addition, a user may request login privileges. From an administrative perspective, a file or database must be available to hold user information. Three classes of users (guest, basic, and super) have been implemented to demonstrate that different capabilities can be associated with different user classes. In our system, a guest user can only display information. Basic users may modify attribute data associated with GIS layers. Super users may modify attribute data and the layers themselves.

Table 2. User, Client, and Administrative Aspects

Client	Server	Administrative
Login	Login page, user authentication	User information
Configure	Provide choices, take responses	Data repositories
Generate	Content, assembly, generation, conversion	Methods to support generation of worlds
Explore	Monitor, log, content	Associate logging with user

Configure: Session Control

After logging in, the user has the opportunity to select the layers to view and other aspects related to configurating the world. Furthermore, the user can request inclusion of information from prior sessions, i.e. any layer or shape that has been previously added. On the server, this information is stored in files that the server reads after a user logs in. This simplifies maintenance because the code does not need to be modified to bring new information to the user. Furthermore, a server does not necessarily need to be brought down in order to make updates to the data. When the server is updated live, interlocks can be implemented to protect against incoherencies during the update time. The user can request any of the layers that are in the GIS layer repository. In addition, the user can select the aerial imagery that that provides the backdrop.

Generate: Synthesize the World

After taking the user information, the requested information is formated and presented to the user for verification. After which, the world is generated. At this point, the server assembles the information and makes whatever conversions are necessary to include the information in the world. Some of the conversion processes are implementation dependent, requiring interface with the client application. Other information, GIS information in particular, requires translation into VRML. As noted above, the GIS information is typically represented as double precision floating point numbers (a precision of about 14 decimal digits), whereas, VRML can only handle single precision (only about 7 decimal digits). Seven decimal digits of precision is insufficient to represent georeferences GIS data. As a result, double precision values have been translated so the new origin is at the center of the requested world. This maintains the precision within the world so that quantization effects are small. The VRML based GIS layers are bundled together and integrated into the requested world. The features included in this layer are described in the next subsection presenting the client architecture. Finally, if a user layer has been defined and requested, the user layer is also inserted into the world.

Explore: Monitor and Respond

Once the client has rendered the world, the user explores the world. At various times, information is passed back to the server to provide a record of certain events, such as the addition of a feature. Also, URLs can be associated with different features that result in requests for documents from the server.

Client Architecture

The details of the client architecture are described elsewhere [5–7]. The client is written entirely in VRML and salient aspects of the architecture are described here. The purpose is to render the world and allow the user the ability to navigate through and interact with the world. In addition to managing the direct interactions with the user, the client architecture must also manage the information provided by the user and also to communicate with the server. The client architecture is shown in Figure 2.

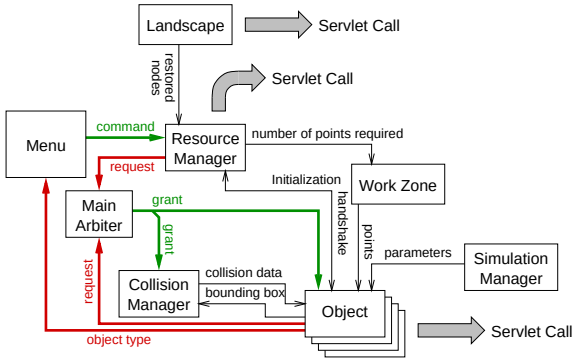


Figure 2. Client Architecture [7]

Base Visualization

The base visualization encompasses the region of interest in the visualization. In our application, this area is nearly flat, so a plane is used to model the terrain. Aerial imagery is painted on the plane to provide the realistic backdrop for the world. The imagery is selectable from the server allowing inclusion of different imagery at different resolutions. We had one foot resolution imagery available, however this resolution resulted in an image that was impractical to include in its entirety. As a result, the image was down sampled by factors of two, four, and eight. In addition, all information was provided in geographic coordinates as double precision floating point numbers. For the actual shapes defined in the VRML world, the origin was moved to a point near the center of the region of interest. The base visualization also includes a programmed fly through that flies over certain strategic points in

the world. The fly-through also displays the names of various points as they are encountered.

GIS Layers

A key difference in the application presented in this paper compared with prior cases [5–7], is the inclusion of GIS layers that are controllable by the user that are derived on demand on the server. After the user logs in, one of several GIS layers may be selected for inclusion. The client application retrieves the GIS layers by instantiating a top level EXTERNPROTO node specified with servlet URL that includes all requested GIS layers. In this way, any number of layers can be included while maintaining a uniform interface with the client application. The individual GIS layers are nested in the top level EXTERNPROTO and are again defined by servlet URLs that direct the server to perform the required conversions. The GIS layers are rendered in the form specified in the shapefile, with polylines implemented as VRML `IndexedLineSets` and polygons implemented as `IndexedFaceSets`. Attribute information appears when the users pointing device (often a mouse) passes over the shape. When the user clicks on the shape, a web page appears with the attribute data. Different user classes are possible, and the user may also have the opportunity to modify the attribute data on the server.

User Interface

To control the different capabilities and features, a user interface is included to help guide the user in the capabilities of the world. The user interface is implemented as a head's up display that always follows the user. As noted above, the world may include several GIS layers. Through the menu, the layers may be hidden and displayed on request. In addition, the user may create specific layers based if desired. The menu is used to initiate and control the process of adding layers. Finally, the user interface includes a navigation panel that assists in determining the location within the world.

Resource Manager

In order to manage live updates in a VRML world, a URL containing VRML must be retrieved and then inserted into the world [7]. The insertion is simple provided it is permanent. If the ability to modify inserted content is required, a resource manager is necessary to manage the insertion of the content. The resource manager maintains tables for all addable object types, all objects that have been added, manages initialization of new objects and connects new content into the world.

Communications with the Server

Browser applications are typically limited in what they are capable of doing to protect the user from malicious or inadvertent modification of data on the client machine. Thus, in order to capture information from a user session, communication with the server is necessary. The client application communicates with the server by requesting URLs for servlets. These servlets may direct the server to log the information or may also be a request for VRML content that can be subsequently be added to the world. These communication issues are described in detail in [7].

EXAMPLE

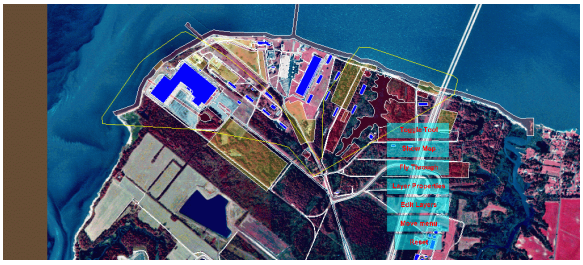
The Former Nansemond Ordnance Depot (FNOD) in Suffolk, Virginia [9] was an ordnance site for World War I and World War II munitions. During this time, various ordnance was discarded and also resulted in some collaborative chemical contamination. An interactive virtual world has been created to provide a way to view the progress of the clean-up, to be simple to use, and simple to maintain. The information provided by the Army Corps has included data defining different clean-up regions, building outlines, and other geographic information provided in the form of ESRI Shape files. In addition, geographically registered aerial imagery has been provided. Recognizing that this is in all likelihood the same information that would be available for any cleanup effort, the application is defined to be largely generic with the GIS information customizing any particular site. As a result, the application is capable of taking any collection of shape files and imagery. Image size is an issue, so several down-sampled versions of the image are offered as choices. In addition, the application integrates multiuser capabilities previously developed [5] that enable users to login and manage a user space, the ability to configure their virtual world, the ability to modify aspects of the world, and a limited ability to collaborate with other users. An example session begins with the user logging into the server. The user, after logging in, is presented with a set of choices for configuring a world. The user may select the layers to show and the imagery to include. From this information, the server generates a world that is navigated in the client. Figure 3 gives some selected images from a session.

SUMMARY

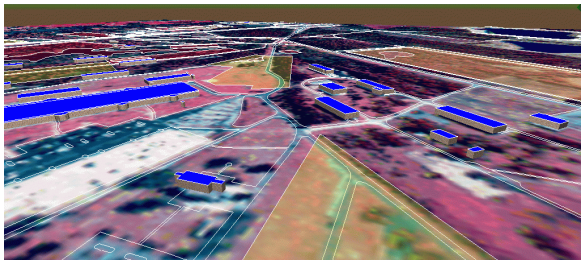
In this paper, we have described an application that can be used to visualize and monitor progress of a SuperFund cleanup site. The application is built upon the ILUVA architecture previously developed by the author. The application features dynamic generation of a world specified by the user. Furthermore, visualizations of the VRML GIS layers are translated on demand from ESRI Shapefiles. The application was described and and example session was presented.

REFERENCES

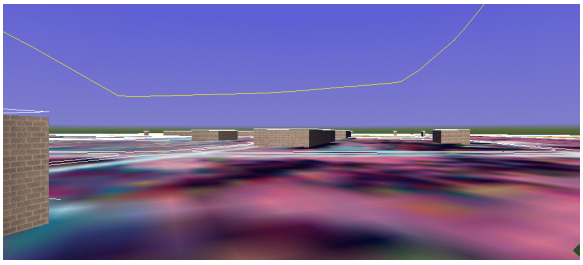
[1] Martin Reddy, Yvan Leclerc, Lee Iverson, and Nat Bletter, "TerraVision II: Visualizing massive terrain



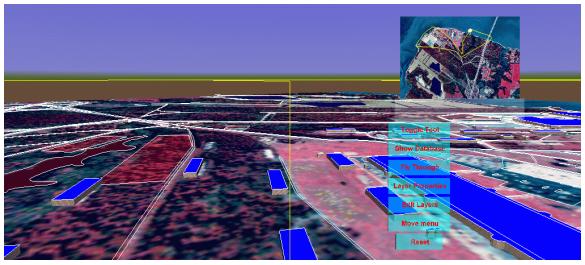
(a) Opening Viewpoint



(b) Aerial Viewpoint



(c) Ground Viewpoint



(d) On Animated Viewpoint

Figure 3. Some images from the FNOD visualization

databases in VRML", *IEEE Computer Graphics & Applications*, vol. 19, no. 2, pp. 30–38, March/April 1999.

[2] Martin Reddy, Lee Iverson, and Yvan G. Leclerc, "Under the hood of GeoVRML 1.0", in *Proceedings of the Fifth Symposium on the Virtual Reality Modeling Language VRML2000*, Monterey, CA, February 2000, pp. 23–38.

[3] Lee A. Belfore II and Rajesh Vennam, "VRML for urban visualization", in *1999 Winter Simulation Conference Proceedings*, Phoenix, Arizona, December 1999, pp. 1454–1459.

[4] Agostino G. Bruzzone and Giovanna Berrino, "Modelling of urban services by VRML & Java", in *Proceedings of the 1999 International Conference on Web-Based Modeling & Simulation*, San Francisco, CA, January 1999, pp. 34–38.

[5] Lee A. Belfore II and Suresh Chitithoti, "Multiuser extensions to the interactive land use VRML application (ILUVA)", in *Thirty-Fourth Annual Simulation Symposium*, Seattle, Washington, April 22-26 2001, pp. 159–166.

[6] Lee A. Belfore II and Suresh Chitithoti, "An interactive land use VRML application (ILUVA) with servlet assist", in *2000 Winter Simulation Conference Proceedings*, Orlando, Florida, December 2000, pp. 1823–1830.

[7] Lee A. Belfore II, "An architecture for creating large VRML worlds", *Transactions of the Society for Computer Simulation*, March 2001, 24-40.

[8] ESRI, *ESRI Shapefile Technical Description*, July 1998.

[9] US Army Corps of Engineers, Norfolk District, *Former Nansemond Ordnance Depod (FNOD), Suffolk Virginia*, `http://www.nao.usace.army.mil/Projects/Nansemond/welcome.html`.

[10] The Web3D Consortium, Incorporated, "The Virtual Reality Modeling Language", `http://www.web3d.org/Specifications/VRML97/index.html`, 1998.

BIOGRAPHY

Lee A. Belfore, II, has been an Assistant Professor in the Department of Electrical and Computer Engineering at Old Dominion University, Norfolk, Virginia, USA, since 1997. He is also an affiliated faculty member with the Virginia, Modeling, Analysis and Simulation Center (VMASC), Suffolk, Virginia. Dr. Belfore received his BS in Electrical Engineering from Virginia Tech, Blacksburg, Virginia, in 1982, his MSE In Electrical Engineering and Computer Science from Princeton University, Princeton, New Jersey, in 1983, and his Ph.D. in Electrical Engineering from the University of Virginia, Charlottesville, Virginia in 1990. His research interests include Internet based virtual reality and data compression.