

A Tool for Visualizing SEDRIS Databases Across the WWW

Suraiya Haque Suliman
Paul J. Metzger
Reality by Design
354 West Cummings Park
Woburn, MA 01801 U.S.A.
781-937-9370, 781-937-9371 (fax)
shaque@rbd.com and pjm@rbd.com

ABSTRACT

The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) project sponsored by the Defense Modeling and Simulation Office (DMSO) and the Army Simulation, Training, and Instrumentation Command (STRICOM) is focused at standardizing the synthetic environment data representation requirements and interchange mechanisms for all networked modeling and simulation systems. To achieve this goal, SEDRIS uses data modeling for lossless interchange of correlated synthetic environment data among the heterogeneous components of Modeling and Simulation systems. In order to promote and facilitate the emerging SEDRIS standard, tools for accessing and visualizing the SEDRIS databases need to be available to the broadest possible segment of the Modeling and Simulation community. One approach is to use the popularity of the Internet and develop a cross platform SEDRIS viewer for a World Wide Web browser. This paper presents a tool for the real-time representation of SEDRIS databases across the World Wide Web. Accessing SEDRIS data using a client/server model as well as support for 3D visual, audio, and gridded data in the viewer will be discussed. Architectural considerations will be presented in support of performance requirements of the viewer as well as the network protocol implemented for data transmission between the client and server.

Overview

When several disparate distributed modeling and simulation systems need to interoperate, a correlated view of the natural synthetic environment is a critical precondition for a "fair fight." In most cases, each system has particular requirements determined by its own unique specifications such as fidelity, performance, visual representation techniques, etc. Traditionally, data correlation between systems has been achieved by the conversion of databases from one format to the other by system-specific conversion utilities. These point-to-point data interchanges are expensive and do not provide a generalized approach to the data conversion problem that subsequent conversions can build on. The SEDRIS project, sponsored by the Defense Modeling and Simulation Office (DMSO), addresses this problem by defining a common, standardized transmittal medium which can serve as the basis of our interchange (Ref. 1). SEDRIS provides a common definition of the synthetic environment by capturing all the data types and their relationships with the use of data models (Ref. 2). This enables the interchange of unambiguous and lossless data between live, virtual, and constructive simulations. Using a polymorphic approach in the data models, SEDRIS is able to support the requirements of heterogeneous M&S systems.

SEDRIS consists of a well-defined application programmer's interface (API) which decouples the internal data structures of the data model from the application programs which generate and access data via SEDRIS. Using this single API, application programs are implicitly capable of supporting a variety of data formats supported by SEDRIS. Applications no longer need to provide explicit support for multiple native data formats. The structures of the various data formats are transparent to the application accessing the data via this API. The SEDRIS Navigator was designed and implemented to visualize various data formats by using SEDRIS. The Navigator does not have any explicit knowledge of the native data formats but utilizes the SEDRIS API to visualize any databases that have been imported to SEDRIS.

The SEDRIS Navigator is composed of a client and a server. The client is a cross-platform OpenGL application that visualizes the database. It is a helper application for

Internet browsers such as Netscape Navigator or Microsoft Internet Explorer. The server is an "application server" which listens for connections from the client applications, accesses data via the SEDRIS API, performs some application-specific filtering, and transmits the resultant data to the requesting client.

Client

The client is the front-end module of the Navigator that visualizes data accessed from a networked server. It provides a user interface that allows a user to view individual models and textures in addition to a complete view of the terrain database. Individual model and texture manipulation as well as database flythrough and teleporting capabilities are provided through keyboard and mouse interactions.

The client is a cross-platform application that has a performance requirement of being able to access and visualize large 3D databases in real-time. OpenGL was chosen as the 3D graphics API due to its cross-platform nature in addition to its computational speed. The implementation of OpenGL supports both software-only and hardware-accelerated modes on PCs (Windows 95 and NT as well as various flavors of Unix), Unix workstations (SGI, Sun, and others), and Macintosh.

The client application is implemented as a helper application. This allows the client to run with any Internet browser, as well as in stand-alone mode. Helper applications are treated in a similar fashion to plug-ins. The difference is that a helper application is a totally separate executable, with all aspects of software execution and memory management under the control of the client. The common purpose of plug-ins and helper applications is to provide application-specific methods for processing data types that are not natively recognized by the browser. The method of identifying the different data types is via the Multipurpose Internet Mail Extensions, or MIME types. In addition to MIME types, file extensions are used as a common way of indicating the type of data to be presented. For the SEDRIS Navigator, we use the mime type of "x-world/x-sedris" and an associated file type of "sedris". The client application and the associated MIME type are registered with the Internet browser residing on the client computer. When a document with the specified MIME type is accessed, the client application is invoked.

The client then attempts to connect to the server. If this succeeds, the client initiates transmission of the desired data in the form of an OpenGL data stream.

As a 3D viewer that accesses databases across a network, the effectiveness of the client is limited by the underlying hardware platform and network bandwidth. The client extends the range of hardware/bandwidth requirements by supporting multiple fidelities in data accessed from the server. This concept of multiple-fidelity support is implemented at two levels in the client. First, the client can control the complexity of the data it requests from the server. For example, a high-end client may request smooth shaded and textured polygons using 64-bit floating-point precision, while an entry-level client (perhaps a laptop with a 14.4k modem) may request simple vertex data with 32-bit precision in order to render a wireframe representation. The other option available is to simplify the rendering only by switching to less computationally-intensive techniques such as wireframes and non-textured mode.

Client Data Caching and Paging

SEDRIS databases can be arbitrarily large requiring many gigabytes of disk space. The client incorporates a terrain paging and caching mechanism (see diagram at the end of this paper) which allows visualization of larger databases than would otherwise be possible with available disk space. At any point in time, the client is only interested in a subset of the database – namely the region in the viewing range of the user. The client defines this region as a square centered at the viewpoint and maintains all the information associated with this region in memory. The sides of the square are twice the distance to the far clipping plane. The client processes data in rectangular patches called tiles. This allows the client to perform some optimized data access and performance-based culling independent of the hierarchy with which the database was created. Using tiles, the client only inserts the polygons within the viewing range into the OpenGL scenegraph for rendering, offloading the lower level graphics pipeline. At startup and as view positions are changed, the client checks whether the data is available in memory or on local disk, in that order, before initiating a connection with the server. As viewpoints move, tiles are paged in and out of memory. If a tile cannot be found in RAM or on disk, a message is sent to the server requesting the tile. On

receipt of the tile, the information is saved on disk and is added to the scenegraph. The amount of memory available for paging and the location of cache files are user-specifiable via registry or environment variables.

Server

The SEDRIS Navigator server is responsible for listening to connections from the client and transmitting requested data to the client accessed via the SEDRIS API. The server does not need to be integrated with a web server. It is a multi-threaded application designed to handle multiple clients simultaneously. The server listens for connections from a client on a predefined TCP/IP port and, on connection, spawns a process to service the request.

The server is the application that has the SEDRIS read API compiled into it. The server must be able to support client connections that request different underlying data formats. One client may request Lockheed Martin S1000 format, while another might request MultiGen OpenFlight. This requires that the server support different instantiations of the SEDRIS read API. The request for S1000 data must go through the S1000->SEDRIS read API, while the OpenFlight request must go through the OpenFlight->SEDRIS read API. SEDRIS provides the various data format libraries as dynamic link libraries (DLLs), making it possible to support multiple data formats in a single server. This (DLL) abstraction mechanism supports derived class format implementations of the SEDRIS API. This approach permits multiple SEDRIS transmittals of different formats to operate simultaneously. In terms of the Navigator, this means that one executable (the server) supports all format implementations of the SEDRIS API. Support for future format implementations can be added by incorporating a new DLL and defining the association between the transmittal filename extension and the DLL name.

Server Data Caching

Server terrain caching utilizes the feature of search boundaries in SEDRIS. This feature allows a user to specify a region of interest for the SEDRIS transmittal. The server maintains on disk a local cache of data already retrieved from the SEDRIS transmittal. When a request for data is received from the client, the server first checks its local cache for availability. If the

data exists in the cache, it is sent directly to the client. If the data is not found in the cache, it requests the data from the SEDRIS transmittal, saves it in cache and transmits it to the client. The location of the cache is specified in a registry or environment variable. In the absence of both, the local directory is used as the cache location (see diagram at the end of this paper).

Client/Server Communication Protocol

The TCP/IP protocol is used for communication between the client and server over the Internet. It is a request/response system designed for communicating data on an "as needed" basis. The client is responsible for evaluating its data needs and initiating all transfers. The client can send 10 message protocol data units (PDU) and the server can send 11 PDUs. Each packet consists of a SEDRIS PDU header consisting of the packet length, packet type, and protocol version. Below is a list of PDUs in control flow order for a session requesting display of terrain.

CLIENT CONNECTION PDU

Client to Server, requesting communication initialization. If local cached data cannot be found, this PDU is sent after the client has initiated the TCP/IP connection with the server. The PDU specifies the fidelity, format and filename of the database. These fields are obtained by processing the contents of the *.sedris file sent by the web server. In addition, the client specifies the desired endian-ness of the data packets (Big Endian or Little Endian). This feature is used to place the burden of byte swapping on the server and allows clients to receive data in their native format. This PDU also contains a "polygon limit" field, primarily used for testing. The server will only send up to the number of polygons specified by this limit.

STATUS RESPONSE PDU

Server to Client, indicating whether the request was successful.

OPEN TRANSMITTAL PDU

Client to Server, requesting initialization of a SEDRIS transmittal. The server maintains a mapping between process ID and SEDRIS transmittal handle to support multiple client connections.

TRANSMITTAL RESPONSE PDU

Server to Client, providing status on whether the transmittal could be opened. If successful, the server also sends back metadata, including database origin, version, creation date, number of models, and number of textures. This packet may also potentially list the names of the models and textures, allowing the client to associate human-readable string names with numeric identifiers.

SPATIAL DOMAIN REQUEST PDU

Client to Server, requesting the extents of the database.

SPATIAL DOMAIN TRANSMITTAL PDU

Server to Client, transmitting the extents of the database by specifying the southwest and northeast corners of the database.

SET FIDELITY PDU

Client to Server, setting the fidelity level of future packets sent from the server.

STATUS RESPONSE PDU

Server to Client, indicating whether the request was successful.

TILE REQUEST PDU

Client to Server, requesting a synthetic environment tile by specifying the extents (southwest and northwest corners) in the PDU. When the server receives this, it starts sending the polygon and geometry model instances contained in the tile. As the client receives the data, if there is an item required (e.g., a texture map) for this object, a request is made to the server. When all data for the object have been received, the client requests the next object. The client enters a processing loop and waits to receive the TILE RESPONSE PDU from the server which signals completion.

GEOMETRY MODEL INSTANCE PDU

Server to Client, indicating the ID, position, and orientation of a geometry model instance.

POLYGON PDU

Server to Client, containing polygon information for the terrain skin in a tile as requested by the client. The packet has variable length fields that hold data appropriate for the fidelity level requested.

TEXTURE MAP REQUEST PDU

Client to Server, requesting a particular texture map by its ID.

TEXTURE MAP RESPONSE PDU

Server to Client, including the checksum of the texture map. The client uses this to

determine if the locally cached version is current.

LOAD TEXTURE MAP PDU

Server to Client, containing the binary image file requested including the texture map ID, name, and the image data. The data is in the SGI Image File format (either RGB or RGBA). It is not run-length encoded.

MODEL REQUEST PDU

Client to Server, requesting a particular geometry model definition by its ID.

OBJECT START PDU

Server to Client, signaling the beginning of a geometry model definition. Contains the object ID, name, and a Boolean flag indicating whether this object is a rotating stamp (such as a tree). This PDU may be nested within other Object Start PDUs, but must always have an associated Object Stop PDU.

POLYGON PDU

Server to Client, containing polygon information for a model requested by the client. The packet has variable length fields that hold data appropriate for the fidelity level requested.

OBJECT STOP PDU

Server to Client, signaling the end of a model.

TILE RESPONSE PDU

Server to Client, indicating the completion of tile transmission.

CLOSE TRANSMITTAL PDU

Client to Server, requesting that the server close the SEDRIS transmittal and free all associated memory.

STATUS RESPONSE PDU

Server to Client, indicating whether the request was successful.

Conclusion

We have successfully visualized numerous databases using the SEDRIS Navigator including Lockheed Martin's S1000, MultiGen's OpenFlight, Evans and Sutherland's CCTT and STF, and the SEDRIS transmittal format. We have extended this capability to our Simulation Visualization System (SVS) (Bar97) by integrating the Navigator client into SVS. This incorporation allows the SVS system to implicitly visualize any database supported by SEDRIS. An SVS client replaces the Navigator client and communicates with the server using the existing protocol.

Several differences between an SVS client and a Navigator client exist. The first difference is in the method of visual system rendering. The Navigator client builds an OpenGL scenegraph and renders the data directly. SVS goes through a layer of middleware and does not make OpenGL calls directly. This middleware (which differs based on the underlying hardware architecture) supports the construction of a scenegraph via a well-defined API.

The second difference is that SVS requires the ability to perform intersection testing with polygons, while the Navigator client does not. When terrain skin polygons are received by SVS, they are incorporated into an "intersection database" as well as a "graphics database". The intersection database is required to support collision detection and response, as well as ballistics flyout. The middleware utilized by SVS supports the creation of this type of intersection database.

For RBD's Dynamic Simulation Environment (DSE) (Bar98) project, the Navigator client was incorporated to perform data import and scenegraph visualization. For applications that need to maintain their own rendering capabilities, the server can be used independently for accessing SEDRIS data via the TCP/IP protocol.

Future Work

Our current paging algorithm is a simple one, loading tiles as they are rendered. A forward-looking algorithm that pre-fetches and pre-loads tiles may reduce wait time and improve performance. Extension of the Navigator client to play audio data obtained via SEDRIS is also underway.

Support for gridded data will be implemented by incorporating algorithms on the server to turn a SEDRIS Property Grid into a set of polygons to be sent to the Client. The types of Property Grids that will be supported initially are two dimensional grids with a single elevation (or depth) value associated with each grid cell. The generated polygons will be assigned colors based on min/max elevation values. As 3D gridded data become available via SEDRIS, it is anticipated that the Navigator will be extended to visualize atmospheric and oceanographic data. The Navigator architecture and real-time data access capability of the client will readily allow visualization of time-variant environmental data with the extension for 3D gridded data.

Sponsors and Further Information

The SEDRIS Navigator was developed with sponsorship from the Defense Modeling and Simulation Office, with support from the U.S. Army's Simulation, Training, and Instrumentation Command and the Naval Air Warfare Center Training System Division.

General information on SEDRIS can be obtained via the Internet at www.sedris.org. Questions regarding SEDRIS or the SEDRIS Navigator should be sent via e-mail to segmt@dss.ll.mit.edu.

Authors

SURAIYA HAQUE SULIMAN is a software engineer at Reality by Design with previous experience from Lockheed Martin Information Systems, Advanced Distributed Simulation Division. She has experience in the areas of synthetic environment protocol development and data visualization, physical model development for real-time systems, and user interface design and implementation. Software research and development areas for military simulation include work with modular semi-autonomous forces (ModSAF), environmental models, and network protocol development for DIS and HLA. Ms. Suliman holds a Bachelor of Science in mathematics from York University, a Master of Science in computer science from the University of Massachusetts, Lowell, and is in the final stages of completing a Doctorate of Science in computer science at the University of Massachusetts, Lowell.

PAUL J. METZGER, a co-founder of Reality by Design, is VP of Engineering at RBD. Mr. Metzger was the technical lead and principal developer of the SEDRIS Navigator. Mr. Metzger also has extensive experience with the High Level Architecture, having developed the HLA RTI-s Management Object Model, ported the HLA RTI-s to Linux, and assisted in the transition of the RBD PC-based HMMWV to the HLA, the very first officially certified HLA compliant federate. At RBD, Mr. Metzger is responsible for the cross platform Soldier Visualization Station™ (SVS™), SVS Stealth™, and SoundStorm 3D™ products. Mr. Metzger has extensive experience in 3D graphics and networking, having worked in distributed simulation since the early days of the SIMNET program.

References

US98-1, US Army's Simulation, Training, and Instrumentation Command, Synthetic Environment Data Representation and Interchange Specification Overview, volume 2 of the SEDRIS Documentation set, 28 March 1998 Orlando, Florida, 12350 Research Parkway, Orlando, FL.

US98-2, US Army's Simulation, Training, and Instrumentation Command, Synthetic Environment Data Representation and Interchange Specification Reference Manual, volume 4 of the SEDRIS Documentation set, 28 March 1998, Orlando, Florida, 12350 Research Parkway, Orlando, FL.

Bar97, Barham, Paul T., Jones, Traci A. "Individual Combatants in Dismounted Warrior Network" in the Proceedings of the 97 Fall Simulation Interoperability Workshop, Orlando, FL, September 8-12, 1997.

Bar98, Barham, Paul T., Barker, Randall E., Metzger, Joanne L., "Dynamic Simulation Environment", to be published in ITEC 1999, The Hague, Netherlands.

Control Flow Diagram of Client and Server

