# *i3D*: A High-Speed 3D Web Browser

Jean-Francis Balaguer and Enrico Gobbetti †

Center for Advanced Studies, Research and Development in Sardinia

## ABSTRACT

In this paper, we present *i3D*, a system that combines the 3D input and high-performance rendering capabilities of high-end virtual reality systems with the data fetching abilities of network browsers. Using a Spaceball, the user can intuitively navigate inside the three-dimensional data, while selecting 3D objects with the mouse triggers requests for access to remote media documents that can be text, still images, animations or even other 3D models. Time-critical rendering techniques allow the system to display complex 3D scenes at high and constant frame rates, making it possible to use it in the context of large scale projects. The system is currently being used at CERN as a visualization and data management tool for the design of the new Large Hadron Collider, and at CRS4 for the Virtual Sardinia project and in the networked educational system *IPERLER*. *i3D* is available through anonymous ftp from various sites on the Internet.

## Keywords

Hypermedia, 3D Visualization, VRML, WWW Browser, View-and-markup Tools.

† CRS4, Scientific Visualization Group, Via Nazario Sauro 10, 09123 Cagliari, Italy.
E-mail: {balaguer|gobbetti}@crs4.it
WWW: http://www.crs4.it/~balaguer
http://www.crs4.it/~gobbetti
i3d info on WWW: http://www.crs4.it/~3diadm

## 1. INTRODUCTION

The World-Wide-Web (WWW) has rapidly become one of the fundamental structures of the Internet. It adds a universal organization to the data made available on the Internet allowing to view all hosts as a unique data source, and to treat all of this data as parts of a single structured document [11]. The HyperText Markup Language used to describe WWW documents has its roots in SGML [17], a format for printed media, and is therefore intrinsically suited for the composition of bidimensional documents composed of textual and pictorial data. Other types of media like digital video and sound are accessible through the invocation of external specialized viewer applications.

The availability, at a relatively low costs, of 3D graphics workstations able to display scenes composed of thousands of polygons at interactive speeds, has made it possible to bring 3D data to the World Wide Web through specialized viewers for this new kind of media. However, it has been rapidly identified that the effectiveness of interactive 3D viewers in communicating information about 3D environments can be dramatically enhanced by attaching digital media annotations to the environment's models. By allowing users to interactively recall and view the attached information by selecting objects of interest during navigation, the interactive 3D viewer becomes a natural front-end for querying information about 3D models. Annotations can refer to text, still images, animations or even other 3D models, exploiting that way all of the digital media capabilities of current workstations.

As an example, in architectural CAD applications, the virtual building representation could be augmented by linking to its various components the original drawings showing engineering details of the structure, photographs of the real site, and so on. The interactive 3D model can therefore be used for data management purposes during the design phase, and information about the building can be presented to the client with maximum efficiency.

When exploring three-dimensional environments, navigation using interactive control of virtual camera motions is often the most important form of three-dimensional interaction [20][6][9]. Multiple degree-of-freedom input devices as the *Spaceball* and the 3D mice allows interactive 3D viewing with continuous viewpoint control, hence providing visual cues of an invaluable help in understanding the structure of the three-dimensional data. It requires that images be rendered smoothly and quickly enough so that an illusion of real-time exploration of a virtual environment can be achieved as the simulated observer navigates through the model [6][2][19].

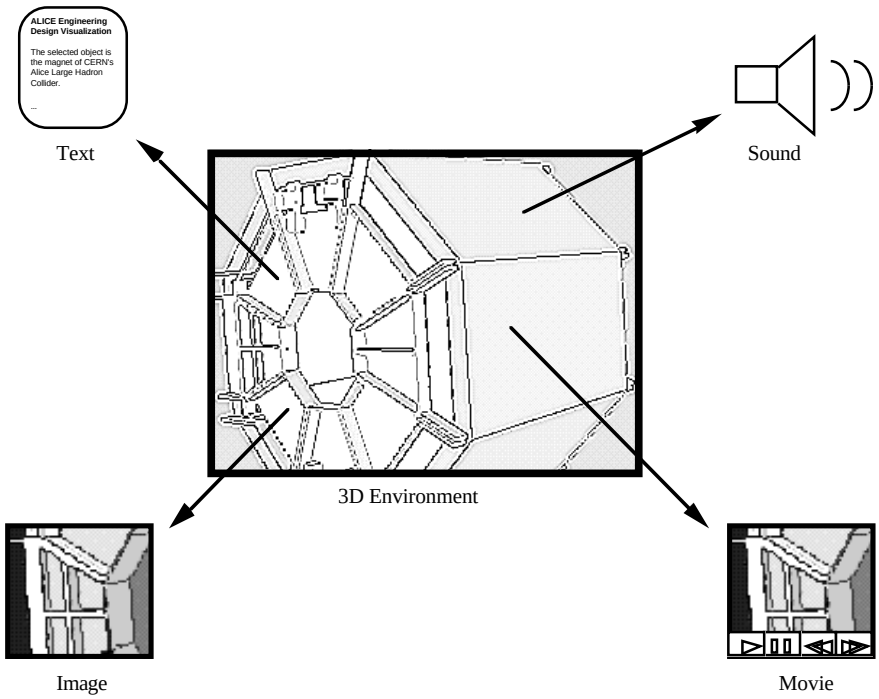


**Figure 1. Annotated 3D Model**

To best perform its function, an interactive system for exploring annotated 3D environments should combine the following characteristics:

- **interactive 3D viewing capabilities**: effective exploration of a 3D environment requires the ability to directly specify 3D motions and the generation of enough depth cues to understand the structure of the 3D world [4];
- **time-critical rendering**: high feedback bandwidth and low response times are crucial for an interactive 3D graphics system, where motion parallax is only obtained by means of high frame rates and continuous motion specification. In order to guarantee low response times when handling large datasets, the system must be able to adaptively trade rendering and computation quality with speed [6];
- **distribution and sharing:** with the World Wide Web, two standard mechanisms for defining distributed documents and sharing information over a network have been introduced: the *Uniform Resource Locator* (*URL*) mechanism, for locating information residing anywhere within the Internet domain, and the *Hypertext Transfer Protocol* (*HTTP*), for rapid file transfer [3]. By using URLs to represent annotations on 3D models and HTTP to fetch the documents upon request, annotated 3D environments can be distributed and shared over the Internet
- **multiple kinds of annotations:** users should be given the possibility to attach to 3D models the kinds of digital media that are best suited to convey their ideas; the 3D system should thus be able to recognize multiple kinds of annotation and to communicate with other viewers capable of handling the various digital media (images, hypertexts, movies); The hypertext transfer protocol HTTP uses the Multipurpose Internet Mail Extensions, which describes a set of mechanisms for specifying and describing the format of Internet message bodies, for request and response message formats. This allows servers to use a standard notation for describing document contents. When a client receives a MIME message, the content is used to invoke the appropriate viewer by analyzing the mailcap configuration file that describes the bindings between document types and viewer applications. That way, it is easy to add local support for a new format without changes to the web browser;

There are a number of systems that currently have some of these capabilities, but none that possesses them all. Visual simulation and virtual reality systems such as *dVISE* [7] and *Performer* [14] strive at providing support for high-performance rendering and multi-dimensional input but do not permit the annotation of 3D models with other digital media. *Iris Annotator* [16] is a 3D view-and-markup utility that allows users to attach digital media annotations to 3D objects. However, *Iris Annotator*'s documents are monolithic and cannot be shared over the network. WWW browsers such as *NCSA Mosaic* or *Netscape* are good at fetching many kinds of data over the Internet, but are currently not able to deal with annotated 3D objects [5]. To overcome this limitation, 3D geometry viewers such as *Geomview* [10] have been connected to WWW browsers to permit the inclusion of 3D models as elements of an hypertext document [1][8], a new language, VRML, has been defined to serve as a standard for defining 3D scenes hyperlinked with the World Wide Web [12][13], and VRML browsers such as WebView from the San Diego Supercomputing Center and Webspace from *Silicon Graphics*, both based on Open Inventor [15], have been developed recently. The integration of a standard network browser with a 3D geometry viewer offers an ideal basis for a system geared towards the exploration of annotated 3D environments, but the geometry viewers that have been integrated with the WWW to date, as well as available 3D browsers, are limited to mouse-based interaction and are not able to ensure constant high frame rates when dealing with large datasets, thus limiting their appropriateness for large scale projects.

In this paper, we present *i3D*, a system for the interactive exploration of annotated 3D models described using VRML or a proprietary file format. It incorporates the 3D input and high-

performance rendering capabilities of high end VR system with the data fetching abilities of network browsers. The system is currently being used at CERN as a visualization and data management tool for the design of the new Large Hadron Collider, and is available through anonymous ftp from various sites on the Internet.

## 2. THE *I3D* SYSTEM

*i3D* is a tool allowing the exploration of three-dimensional scenes annotated with any kind of media documents that can be accessed on the World Wide Web. It is implemented on top of *X11* and *OpenGL*, and runs on *Silicon Graphics* workstations. Using a 3D device, the user can explore its three-dimensional data and request access to other documents. When retrieving and displaying media documents, *i3D* handles directly the three-dimensional data and collaborates with *NCSA Mosaic* or *Netscape* for other types of media.

### 2.1 Application Overview

As shown in figure 2, *i3D* is composed of the following units:

- **the user manager** is responsible for sensing and analyzing the user's movements and actions in order to recompute the new viewpoint position and orientation, to trigger retrieval of media documents by the protocol manager, and to navigate among the stack of worlds that is maintained by the database manager;
- **the protocol manager** is responsible for the retrieval of media documents from the World Wide Web. Three-dimensional scenes as well as inlined worlds and textures are loaded locally and transmitted to the database manager, while requests for other types of media documents are delegated to a WWW browser (*NCSA Mosaic* or *Netscape*) for retrieval and display by the most adequate viewer application;
- **the database manager** maintains the state of the 3D scenes in order to provide the necessary geometrical information and visual attributes for the user and rendering managers to perform their tasks. It also maintains a stack of the scenes that have been visited to reach the current world and provides fast world switching upon user manager's requests;
- **the rendering manager** is responsible for the generation of the visual representation of the current scene at a high and constant frame rate.

### 2.2 User Interaction

*i3D*'s device configuration uses a *Spaceball* and a mouse as input devices. The *Spaceball* is used for the continuous specification of the camera's position and orientation, while the mouse is used to select objects and access media documents. Both user's hands can therefore be employed simultaneously to input information (figure 3). Additionally, abstractions of the 2D mouse motions into 3D transformations are also provided so that navigation be possible when no *Spaceball* is available. A pop-up menu as well as keyboard commands are used to control various visibility flags and rendering modes. The ability to continuously specify complex camera motions in an intuitive way together with high visual feedback rates provides an accurate simulation of motion parallax, one of the most important depth cues when dealing with large environments [2].

**Figure 3. *i3D*'s device configuration**

To explore three-dimensional worlds, the user can either free fly using an eye-in-hand metaphor [21] or inspect the scene or the currently selected object using a slide-on-ball allowing the camera to rotate around its interest point, placed at the center of the object's (or scene's) bounding-box.

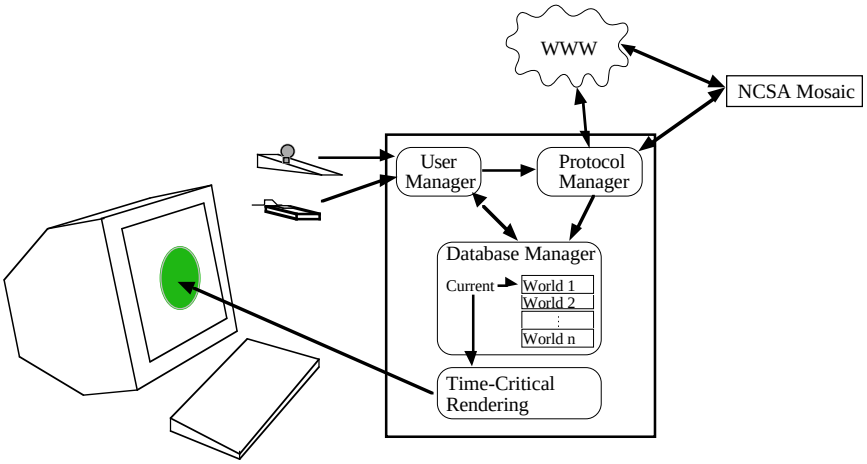


**Figure 2. Application overview**

While navigating inside a three-dimensional scene, the user can request additional information by accessing media documents associated with geometrical data. Since annotated geometries are drawn with a blue silhouette, they can be easily identified. Selecting an annotated geometry by clicking on its visual representation with the mouse triggers the document retrieval and display. For three-dimensional scenes, *i3D* maintains a stack of active worlds. Using keyboard commands, the previous or next world in the stack can be made current, thus providing a mean to quickly navigate among active worlds.

## 2.3 World Model

*i3D*'s database manager stores the representation of three-dimensional scenes as a collection of 3D objects, including light sources and cameras. From the set of world's objects, the database manager builds an octree spatial subdivision to be able to answer rapidly to spatial queries, as for example when selecting objects by tracing a ray from the mouse position.

The generic 3D object possesses the following attributes:

- a **3D transformation** that defines the object's spatial position, orientation and scaling;
- a list of **geometries** representing the different levels of complexity that can be used to render the object;
- a **material**, that defines the way the object behaves with respect to lightning;
- a **texture**, defining the image to be mapped on the object's geometries;
- an **URL**, that locates the media document associated with the 3D object.

Geometric, material and texture objects can be shared by multiple 3D objects of a same scene. Lights have an additional attribute that defines the spread angle while cameras define their viewing volume. The fact that geometrical objects can be associated to lights and cameras allows to make these special types of 3D objects visible and selectable and thus annotatable with media documents. Currently, geometric objects can be described as a list of independent triangles or as a list of independent line segments.

## 2.4 Rendering

The task of the *i3D*'s rendering manager is to display a visual representation of the current world at high and constant frame rates. During navigation, the rendering manager is activated at regular intervals by the main *i3D* event loop and is requested to refresh the screen while adhering to the user-specified timing constraint.

At each activation, the rendering manager renders a single frame by executing the following steps:

- **visibility determination**: first, the database is traversed and the objects visible from the observer's viewpoint are identified. This task is accelerated by first hierarchically determining the visibility of portions of the scene through a traversal of the spatial subdivision maintained by the database manager;
- **display list construction**: each of the objects identified in the previous step is then compiled into a graphical description by stripping off its appearance attributes and compiling them into a device-dependent sequence of commands (in the current version of *i3D*, *OpenGL* display lists are used for storing the compiled versions of graphical objects). During this conversion, geometries are optimized to reduce their rendering time (in particular, structured triangular meshes are generated from the triangle lists stored in the database). To avoid recreating compiled versions at each frame, as it is done in systems like *Performer* [14], *i3D* caches the graphical descriptions generated for each database object and reuses them until they become invalid. The validity of a description is checked by using a time-stamping scheme: the database manager maintains for each of the attributes of the model a time-stamp that is updated every time the attribute is modified, and graphical descriptions are time-stamped by the rendering manager as the most recently modified attribute they requested;
- **level of detail selection**: to reduce the number of polygons rendered in each frame, so as to be able to meet the timing requirements, the rendering manager traverses the generated display list and selects the level of detail at which each of the object will be represented. Level of detail selection is based on the importance of each object for the current frame (which is determined by computing an approximation of its size projected on the screen) and on feedback regarding the time required to render previous frames. The feedback algorithm is similar to the one presented in [14];
- **display list optimization**: once the levels of details are selected, the system has all the information required to render the frame. To exploit coherence, the display list is sorted to optimize the rendering speed; in particular, objects sharing the same texture and/or material are grouped together; in other visual simulation systems, this task is left to the user, that has to encode this information together with the scene description [14][7][15];
- **display list rendering**: the sorted display list is finally traversed and rendered by executing each of the compiled command sequences. Rendering statistics for the current frame are updated and stored so as to be used when selecting the level of detail selection for the next frame.

## 2.5 Utilities

*i3D* uses a specific ASCII file format for the description of the 3D objects and media documents references that composes a three-dimensional scene. 3D objects and their associated visual attributes are considered to be designed externally using modeling tools or CAD systems and then converted to *i3D*'s file format. In order to make scene conversion an automatic process, we provide a suite of converting and scene processing tools that help world designers to import their three-dimensional data into *i3D*.

VRML files are converted on the fly by the browser when VRML are retrieved, either for inlining or when following hyperlinks.

### 2.5.1 Conversion tools

Currently, we provide converters from *Wavefront*'s OBJ ASCII file format and from the subset of the Inventor ASCII file format that defines VRML [12]. For both file formats, only geometries defined by primitives and polygon or line lists are supported. Three-dimensional transformations, as well as material and texture information associated with the converted geometries are also translated when specified. In order to define a complete 3D scene for *i3D*, these individual geometries need to be combined into a single file, media documents must be associated with geometries, and finally the shining lights as well as the rendering

camera must be specified. A specific utility able to process all VRML or *Wavefront* geometry files in a directory and to merge them into a single *i3D* scene file is provided to simplify this step. For VRML files, cameras, lights, levels of detail and media annotations are also converted. If only geometrical data is available, the scene file may need to be edited with a text editor to associate media documents with the geometries and to define the camera and lights. However, *i3D* enforces default behaviors for viewing and lightning when no lights and/or camera are defined in the scene file.

### 2.5.2 Scene Processing Tools

When dealing with geometries, the way objects were constructed or exported with the modeling tool or CAD system may influence the visual representation of the objects or the viewer's rendering performance. A typical problem is the normal direction not being specified or defined only per face, hence resulting in a flat shading rendering of the object. Another typical problem is that often multiple topologically separated models are merged inside a single geometry file, hence preventing efficient culling when rendering. Filtering tools are provided to help world designers to work around these problems by smoothing normal vectors and splitting geometries. Additionally, another filter can be used to define trivial levels of detail by creating oriented bounding-box geometries around small objects.

## 3. APPLICATION EXAMPLES

### 3.1 A Simple Example

In order to give a flavor of defining annotated three-dimensional scenes using the *i3D* file format, figure 5 depicts a file that, when loaded, defines the simple three-dimensional scene shown in figure 4. The scene is composed of three textured cubes. The material and the triangle list defining the cube's geometry are shared among the 3D objects. Each cube is associated with a different texture image that is accessed through HTTP. Media annotations link the cubes to three documents: the cube on the left references an HTML document, the center cube references a MPEG movie, while the cube on the right references an alternate 3D scene.


**Figure 4. A simple annotated 3D scene**

```
Material white_emissive {
         Ambient   0. 0. 0.
         Diffuse   0. 0. 0.
         Emission  1. 1. 1.
         Specular  0. 0. 0.
         Shininess 0.
         Alpha     1.
}
Material red_plastic {
         Diffuse   0.8 0.0 0.0
         Specular  0.2 0.2 0.2
         Shininess 0.5
         Alpha     1.
}
TriangleList unit_cube {
         Triangle p  -1.  1. -1.    -1.  1.  1.     1.  1.  1. \
                  t  -1.  1.  0.    -1.  1.  0.     1.  1.  0.
         …
         Triangle p  -1.  1.  1.     1. -1.  1.     1.  1.  1. \
                  t  -1.  1.  0.     1. -1.  0.     1.  1.  0.
}
Texture skull {
         Image     "http://www.crs4.it/~3diadm/3d/skull.rgb.Z"
}
Node movie_node {
         #
Scale(sx,sy,sz)*Shear(sxy,sxz,syz)*Rot(rx,ry,rz)*Trans(tx,ty,tz)
         frame s    8. 10. 10. \
               t    0.  0.  0.
         material  red_plastic
         texture   skull
         geometry  unit_cube
         url       "http://www.crs4.it/Animate/boneopaque.mpg"
}
Texture sciviz {
         Image     "http://www.crs4.it/~3diadm/3d/sciviz.rgb.Z"
}
Node sciviz_node {
         frame s    8. 10. 10. \
               t  -17.  0.  0.
         material  red_plastic
         texture   sciviz
         geometry  unit_cube
         url       "http://www.crs4.it/~zip/group_homepage.html"
}
Texture alice {
         Image     "http://www.crs4.it/~3diadm/3d/alice.rgb.Z"
}
Node alice_node {
         frame s    8. 10. 10. \
               t   17.  0.  0.
         material  red_plastic
         texture   alice
         geometry  unit_cube
         url       "http://www.crs4.it/~3diadm/3d/alice.3d.Z"
}
Light light {
         frame    t 0. 0. 200.
         material  white_emissive
         angle     30.
}
Camera default_camera {
         frame     t 0. 5. 85.
         perspective 45. 1. 10. 100000. 0.
}
```

**Figure 5. *i3D* textual definition for a simple annotated 3D scene**

### 3.2 The CERN VENUS Project

The European Laboratory for Particle Physics (CERN) is currently involved in designing its next generation particle accelerator, namely the Large Hadron Collider (LHC). In any project of this scale, the design phase is probably the most delicate one, as this is when some critical choices are to be taken which might dramatically affect the final results, timing and costs. The ability to visualize the model in depth is essential to a good understanding of the interrelationships between the parts. An iterative design optimization process can improve remarkably space management and ergonomic issues. However with the visual capability of the present CAD tools, it takes a fair amount of time and imagination to isolate eventual design faults. A pilot project was started at CERN in January '94, named VENUS (Virtual Environment Navigation in the Underground Sites). Its mandate is to produce a detailed *virtual* prototype of the LHC premises and allow navigation and access to engineering data in the form of flythrough by natural interaction. The *i3D* system is being actively used for the exploration of the virtual prototype. Figure 6 shows a snapshot of a typical *i3D* session.

The VENUS virtual prototype is entirely extracted from the original *EUCLID* CAD database. As soon as engineers add new drawings, these are extracted and converted to the *i3D* format in two steps: first, *EUCLID* data is converted to *Wavefront* OBJ format; then, the resulting *Wavefront* objects are converted to *i3D* and assembled into a scene. Some minor manual treatment is necessary at the moment, in order to compensate for lack of some features (e.g. color and textures) in the *EUCLID* to *Wavefront* converter utility supplied by *Matra-Datavision*. Annotations that refer to various sources of information are then added by associating URLs to relevant 3D objects. A further step is necessary to optimize the geometry for interactive navigation. All the process is fairly automatic and does not require major efforts, since the data conversion and optimization are handled by software utilities. The entire conversion process should be completely automated in the near future, and triggered from the *EUCLID* side upon any significant changes to the geometry. This way the virtual prototype will always reflect the latest state of the design.

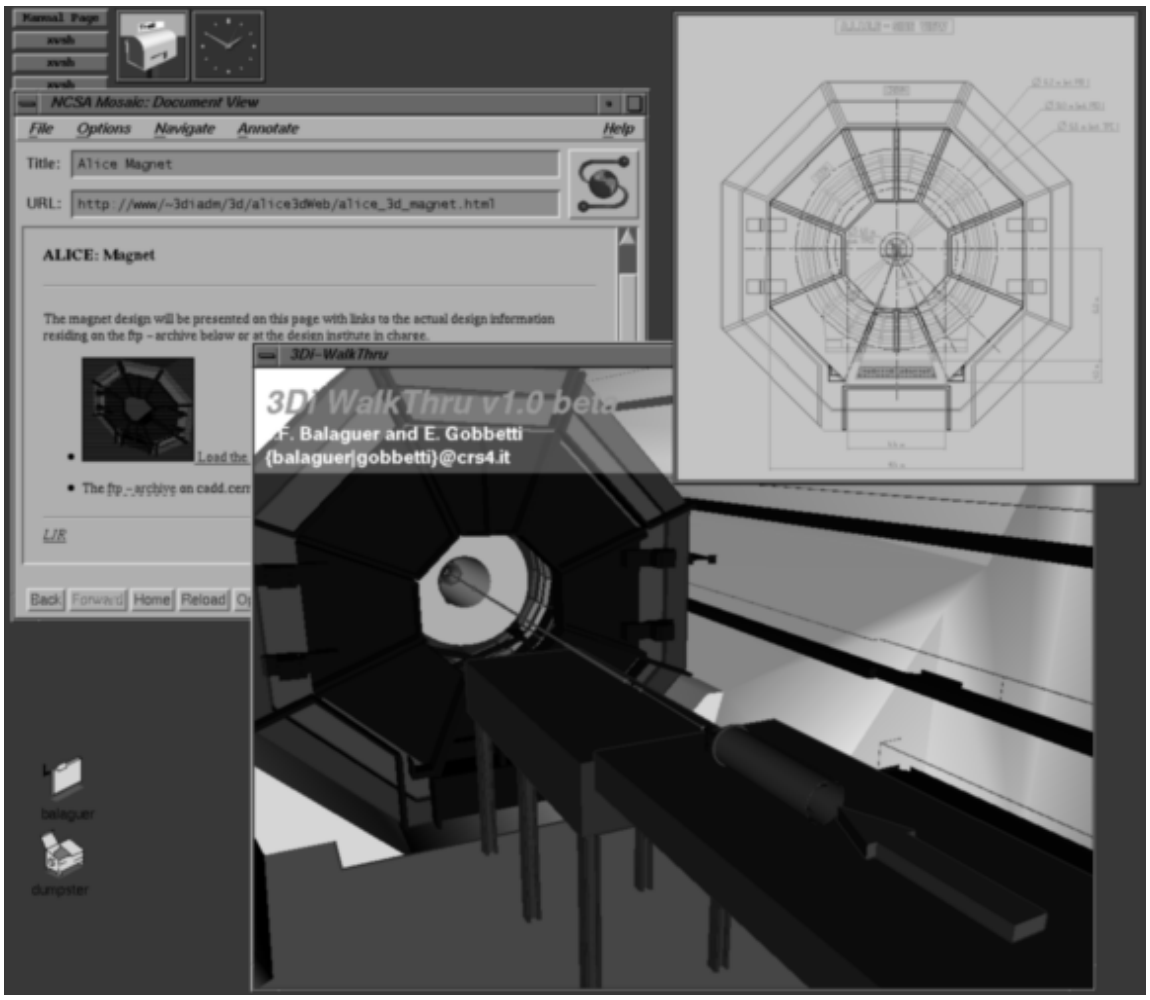


**Figure 6. Exploring CERN Alice**

| RENDERING OPTIMIZATION Target frame rate is 40 ms | RENDERED OBJECTS | RENDERED TRIANGLES | RENDERING TIME |
|---|---|---|---|
| **None** | 171 | 34291 | 80 ms |
| **Culling** | 124 | 20627 | 60 ms |
| **LOD Selection** | 144 | 23843 | 60 ms |
| **Culling and LOD Selection** | 98 | 14419 | 40 ms |

**Figure 7. *i3D* rendering statistics**

*i3D* is made available to all CERN users. This allows any CERN user with a *Silicon Graphics* workstation to connect to the CERN server, to fly through the latest models of the detectors, and to inspect all the information attached to the three-dimensional model (latest CAD drawings, technical papers).

Figure 7 presents the average rendering statistics of *i3D* for the generation of the frame presented in figure 6 on a *Silicon Graphics Onyx RE2*..

Objects in the scene are specified at one or two levels of detail. The user specified an expected frame rate of 20 frames/second and a desired graphical load of 80%. This means that the user desired a constant frame duration of 50 ms, with 40 ms allocated to rendering and 10 ms for all other tasks. By combining hierarchical culling and level of detail selection, the system is able to adhere to the timing requirements.

### 3.3 The Virtual Sardinia Project

The Virtual Sardinia project under realization at CRS4 aims at providing on the Internet easily accessible information on the island of Sardinia.

Using *i3D*, users can explore a 3D model of the island, built from digital terrain model data textured with satellite images. 3D markers are positioned on the surface of the terrain to indicate sites of interest. The interactive selection of one of these markers during navigation triggers the request for accessing a descriptive document, while the selection of a location on the terrain triggers the request to view a high-resolution version of the area surrounding the selected point.

The possibilities of *i3D* are exploited to allow the exploration of detailed terrain models on a range of machines. To produce the *i3D* description of the terrain, the original terrain data (a regular grid) is subdivided into subregions that can be drawn and culled independently, and each subregion is described at various level of detail by transforming the original regular grid into simpler irregular triangular meshes through a decimation process that iteratively groups nearly coplanar polygons and simplifies them. Different tolerances for planarit checks are used to produce the various levels of detail. To avoid cracking problems, small tolerances are used at the borders of the subregions. In addition to the optimization of the geometric model, images that are to be used as textures are clipped and rescaled so as to have them fit into texture memory. All these optimizations are done automatically by a tool that takes as input the original digital terrain model, the satellite images and a list of descriptions of geographical location that have to be marked and assiciated to an hyperlink.

Thanks to hierarchical culling and on-the-fly level of detail selection, the resulting model can be explored at interactive speed (more than 10 frames per second) on a Silicon Graphics Onyx RE2).

The Virtual Sardinia project will be described in details in a forthcoming paper.

## 4. CONCLUSIONS AND FUTURE WORK

We have presented *i3D* a system that combines the 3D input and high-performance rendering of high-end VR systems with data fetching abilities of network browsers. Using a Spaceball, the user can intuitively navigate inside the three-dimensional data, while selecting 3D objects with the mouse triggers requests for access to remote media documents. Time-critical rendering techniques allow the system to display complex 3D scenes at high and constant frame rates. The system is currently being used at CERN as a visualization and data management tool for the design of the new Large Hadron Collider, and will be used at CRS4 in the networked educational system *IPERLER* [18]. Starting from June 1995, unsupported binaries of *i3D* have been also made publicly available by *CRS4* through anonymous ftp at the address *ftp://champagne.crs4.it/pub/I3D*. Mirror sites are listed in *http://www.crs4.it/~3diadm/i3d-install.html*.

Future work will concentrate on improving the interaction with the system by providing additional navigation metaphors and by adding the capability to annotate the 3D scene interactively. We find it also necessary to widen the range of supported file format for data importation. We also plan to improve the visual cues for media annotations so that the user can quickly identify the type of the annotation and determine whether it has already been accessed.

## REFERENCES


[1] Burchard P (1995) *The Cyberview 3D Manual*. The Geometry Center. University of Minnesota.

[2] Brooks FP Jr, Frederick P (1986) Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings *Proc. SIGGRAPH Workshop on Interactive 3D Graphics*: 9-22.

[3] Berners-Lee TJ, Cailliau R, Groff JF, Pollermann B (1992) World-Wide Web: The Information Universe. In *Electronic Networking: Research, Applications, and Policy* 2(1): 52-58.

[4] Bryson S, Pausch R, Robinett W, van Dam A (1993), *Implementing Virtual Reality*, SIGGRAPH Course Notes 43.

[5] Fox E (1993) Digital Libraries. *IEEE Computer*. Nov.

[6] Funkhouser TA, Séquin CH (1994) Adaptive Display Algorithms for Interactive Frame Rates During Visualization of Complex Virtual Environments. *Proc. SIGGRAPH*: 247-254.

[7] Ghee S, Naughton-Green J (1994) Programming Virtual Worlds. *SIGGRAPH Tutorial Notes on Programming Virtual Worlds*: 6.1-6.58.

[8] Munzner T (1995) *An Experiment in Three-Dimensional Distributed Hypermedia*. The Geometry Center. University of Minnesota.

[9] Mackinlay JD, Card S, Robertson G (1990) Rapid Controlled Movement Through a Virtual 3D Workspace, *Computer Graphics* 24(4) : 171-176.

[10] Phillips M, Levy S, Munzner T (1994) *Geomview User Manual*. The Geometry Center. University of Minnesota.

[11] Pesce M, Kennard P, Parise A (1994) Cyberspace, *WWW1 Conference Proceedings*, May 1994.

[12] Parisi A, Pesce M (1994) Virtual Reality Markup Language (VRML). URL: http://www.wired.com/vrml/.

[13] Raggett D (1994) Extending WWW to Support Platform-independent Virtual Reality. URL: http://www.wired.com/vrml/.

[14] Rohlf J, Helman J (1994) Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. *Proc. SIGGRAPH*: 381-395.

[15] Strauss PS, Carey R (1992) An Object-Oriented 3D Graphics Toolkit. *Proc. SIGGRAPH*: 341-347.

[16] Silicon Graphics Inc. (1995) *Iris Annotator User Manual*.

[17] ISO 8879:1986, Information Processing Text and Office Systems Standard Generalized Markup Language (SGML).

[18] Salis C, Leone A (1995) IPERLER Project: Hypermedia and networked Blackboard. Submitted for publication.

[19] Upson C, Fulhauber T, Kamins D, Laidlaw D, Schlegel D, Vroom J, Gurwitz R, van Dam A (1989) The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications* 9(4): 30-42.

[20] Watson V (1989) A Breakthrough for Experiencing and Understanding Simulated Physics. *SIGGRAPH Course Notes on State of the Art in Data Visualization*: IV-26 - IV-32.

[21] Ware C, Osborne S (1990) Exploration and Virtual Camera Control in Virtual Three Dimensional Environments *Proc. SIGGRAPH Workshop on Interactive 3D Graphics* : 175-183.