A Collaborative VolumeViewer

A SMARTDOC IST-2000 28137 prototype

Staffan Palmberg & Magnus Ranlöf

2002-02-26
This study has been carried out as a part of the EC funded project, SMARTDOC IST-2000-28137, with the objective of developing application components that provide highly interactive visualization and collaboration functionalities. The low-level components from the graphics library AVS OpenViz 2.0 are used as the development basis. The application components can be inserted into electronic documents that allow embedded controls such as web documents or Microsoft Word or PowerPoint documents. Instead of displaying results as static images, a SMARTDOC component provides the ability to visualize data and interact with it inside the document.

Although the principal goal of the SMARTDOC project is to create components in a number of different application domains this study concentrates on developing a medical imaging application component in collaboration with the project partners AETmed and professor Alan Jackson at the University of Manchester. By incorporating the application component into patient reports, the clinicians are provided the ability to interact with the 3D data that is described in the reports. To improve the usability of the component, it makes use of a visual user interface (VUI), which gives the user the ability to interact and change parameters directly in the visualization process.

Collaborative work over geographical distances is an area that is becoming increasingly common and thus more interesting. As the availability of bandwidth has increased and the communication technologies have advanced, many companies express their interest for this new practical method of work. A company with offices in different countries would benefit from collaborative techniques providing closer cooperation within the company. Specialized institutions and laboratories could gather much experience and information through collaborative research.

Medical imaging and visualization technique are areas where distinct disciplines such as networking, user interfaces and 3D visualization naturally can be fused together in order to develop collaborative environments. The visualization components developed within the SMARTDOC project will be the foundation for collaborative application components integrated with the Microsoft DirectX® multimedia library. In the medical imaging area, collaborative work can be used to improve diagnoses, journaling and teaching.

This study focuses on developing a prototype of an interactive visualization component for 3D medical imaging and creating a collaborative environment using a multimedia library originally meant for network gaming.
A Collaborative VolumeViewer
A SMARTDOC IST-2000 28137 prototype

Staffan Palmberg & Magnus Ranlöf

Examensarbete utfört i medieteknik
vid Linköpings Tekniska Högskola, Campus Norrköping

Handledare och Examinator: Mikael Jern

Norrköping 2002-02-26
Abstract

This study has been carried out as a part of the EC funded project, SMARTDOC IST-2000-28137, with the objective of developing application components that provide highly interactive visualization and collaboration functionalities. The low-level components from the graphics library AVS OpenViz 2.0 are used as the development basis. The application components can be inserted into electronic documents that allow embedded controls such as web documents or Microsoft Word or PowerPoint documents. Instead of displaying results as static images, a SMARTDOC component provides the ability to visualize data and interact with it inside the document.

Although the principal goal of the SMARTDOC project is to create components in a number of different application domains this study concentrates on developing a medical imaging application component in collaboration with the project partners AETmed and professor Alan Jackson at the University of Manchester. By incorporating the application component into patient reports, the clinicians are provided the ability to interact with the 3D data that is described in the reports. To improve the usability of the component, it makes use of a visual user interface (VUI), which gives the user the ability to interact and change parameters directly in the visualization process.

Digital Imaging and Communications in Medicine (DICOM) is a standard method for transferring images and associated information between different medical devices manufactured by various vendors. Since DICOM has become the de facto standard for management of medical images it is very important that the medical application components are DICOM compatible. The application component developed in this project has been merged with DICOM software from AETmed.

Collaborative work over geographical distances is an area that is becoming increasingly common and thus more interesting. As the availability of bandwidth has increased and the communication technologies have advanced, many companies express their interest for this new practical method of work. A company with offices in different countries would benefit from collaborative techniques providing closer cooperation within the company. Specialized institutions and laboratories could gather much experience and information through collaborative research.

Medical imaging and visualization technique are areas where distinct disciplines such as networking, user interfaces and 3D visualization naturally can be fused together in order to develop collaborative environments. The visualization components developed within the SMARTDOC project will be the foundation for collaborative application components integrated with the Microsoft DirectX® multimedia library. In the medical imaging area, collaborative work can be used to improve diagnoses, journaling and teaching.

This study focuses on developing a prototype of an interactive visualization component for 3D medical imaging and creating a collaborative environment using a multimedia library originally meant for network gaming.
Table of contents

1. INTRODUCTION ........................................................................................................ 3

2. INNOVATIONS ........................................................................................................... 6
   2.1. THE SMARTDOC CONCEPT ................................................................. 6
   2.2. VISUAL USER INTERFACE ................................................................. 6
   2.3. COLLABORATIVE ENVIRONMENT ......................................................... 6
   2.4. 3D ANNOTATIONS ..................................................................................... 7
   2.5. COMPONENT INTEROPERATION ............................................................ 7

3. TOOLS ....................................................................................................................... 8
   3.1. MICROSOFT VISUAL BASIC ................................................................. 8
   3.2. AVS OPENVIZ ......................................................................................... 9
       3.2.1. Component Hierarchy ................................................................... 9
       3.2.2. Scene Tree Architecture ................................................................. 9
       3.2.3. Attributes ...................................................................................... 10
       3.2.4. Interactivity .................................................................................. 10
   3.3. MICROSOFT DIRECTX ............................................................................ 11
       3.3.1. DirectPlay Peer-to-Peer ................................................................. 11
   3.4. DICOM ........................................................................................................... 13
       3.4.1. DICOM architecture .................................................................... 13

4. USER REQUIREMENTS ............................................................................................ 15

5. IMPLEMENTATION .................................................................................................. 17
   5.1. USER INTERFACE ..................................................................................... 18
       5.1.1. Orthoplanes .................................................................................. 19
       5.1.2. Cropping ....................................................................................... 19
       5.1.3. 3D Light ....................................................................................... 19
       5.1.4. Orientation indicator .................................................................... 20
       5.1.5. Resizing of the application component window ......................... 20
   5.2. VISUALIZATION COMPONENTS .......................................................... 21
       5.2.1. Isosurface ...................................................................................... 21
       5.2.2. Orthoslices .................................................................................. 22
       5.2.3. Annotations .................................................................................. 23
       5.2.4. VolumeViewer scene tree .............................................................. 24
   5.3. READ AND WRITE COMPONENTS ......................................................... 25
       5.3.1. Raw data reader ........................................................................... 25
       5.3.2. DICOM reader ............................................................................. 25
       5.3.3. Storing of parameters ................................................................. 26
   5.4. ACTIVEX IMPLEMENTATION ............................................................... 26
   5.5. COLLABORATIVE ENVIRONMENT ....................................................... 27
       5.5.1. Architecture .................................................................................. 27
       5.5.2. Active and Passive Members – Collaborative Rules .................. 28
       5.5.3. Messages ....................................................................................... 29
       5.5.4. Information Protection ................................................................. 31

6. COMPARISON .......................................................................................................... 32
   6.1. APPLICATION COMPARISON ............................................................... 32
       6.1.1. Amira 2.3 ....................................................................................... 32
1. Introduction

VRML (Virtual Reality Modeling Language) was released in the mid 90’s as a standard file format for describing interactive 3D objects and worlds. It introduced new interactive visualization possibilities on the web. The users installed viewers on their machines to be able to display the contents of a VRML file. Despite many advantages of VRML, it is limited and new techniques were invented in order to meet the increasing demands of interactive web graphics. The main limitation of VRML is the amount of data that has to be downloaded to be able to view a file.

The component-based technology offers, for example, client-based methods to present and render interactive 3D graphics as opposite to VRML. It is a clever way of reusing programming code and very advanced and efficient components can be created that meet the tough demands on performance from the users. Many of the low-level components can be reused, since they can be saved on the client side.

The partly EC funded research project SMARTDOC IST-2000-28137 aims at using component technology to develop application components that incorporate interactive visualizations into electronic documents. The partners of the project: Advanced Visual Systems (AVS, Denmark), AETmed (Italy), Intecs (Italy), Unilever (United Kingdom), University of Manchester (United Kingdom) and the University of Linköping (Sweden) have different areas of interest that span from knowledge discovery and engineering to medical imaging. Unlike traditional static images, SMARTDOC application components provide the user the ability to interact with datasets and visualize them in different ways from time to time. These components are new tools for examining various sets of complex 3D data as well as for interactive presentations and reports.

Figure 1 depicts how different low-level components are combined to form SMARTDOC components. The components can be viewed in a SMARTDOC viewer, which can be inserted into a document that allows embedded components, e.g. Microsoft Word. The idea is that the SMARTDOC viewer only has to be downloaded and installed once. It is possible to view different application components in it once this has been done. These components are typically small (some 200 kB).
This report describes how a volume visualization component for medical imaging has been developed. The visualization component has been built to meet the requirements of the end user and the requirements of the SMARTDOC project. One of the members of the SMARTDOC consortium, Prof. Alan Jackson at the University of Manchester, has gathered the end user requirements for an application component. The purpose of the work has been to design, develop and verify an application component, VolumeViewer, which fulfils these requirements. VolumeViewer is primarily meant for investigating medical datasets produced by Magnetic Resonance Imaging (MRI) and Computed Tomography (CT), but can also be used for other datasets. The application component provides integrated 2D and 3D interactive data visualization techniques.

The VolumeViewer prototype has been developed to address three different scenarios:

1. Diagnosis – Stand-alone version, VolumeViewer
2. Collaborative analysis – Network version, VolumeViewerNet
3. Presentation and journaling – ActiveX component, VolumeViewerX

VolumeViewerX can be integrated into different electronic documents while VolumeViewerNet includes a collaborative environment where two users can work together displaying and interacting with the same dataset remotely. The collaborative environment has been developed using the Microsoft DirectPlay® classes of the Microsoft DirectX 8.0a multimedia class library, which is meant for network gaming.

Of central importance to the architecture of SMARTDOC components is the visual user interface, which allows the user to directly manipulate the rendered objects in the visualization. A VUI complements the traditional GUI in order to give the user a more active role in the visualization process.

Chapter 2 contains a presentation of the innovations of the work. In chapter 3, the different tools used to develop the components are described. The requirements of the project are presented in chapter 4. Chapter 5 contains a description of the implementation of the visualization components VolumeViewer, VolumeViewerNet and VolumeViewerX, while chapter 6 contains a comparison.
to other similar software packages and collaborative solutions. Chapter 7 presents
the assessment work that has been done to improve the functionality and usability
of the VolumeViewer versions. Chapter 8 contains conclusions of the study and a
discussion of the results.
2. Innovations

This chapter describes the innovations of the study. The issues presented here will be discussed in chapter 5 where the implementation phase of the study is described.

2.1. The SMARTDOC concept

The innovation in SMARTDOC can be summarised as the integration of electronic documents and embedded 2D and 3D visualization tools based on a low-level, “atomic” component infrastructure [7][8]. SMARTDOC application components will share an “engine”, responsible for all rendering and interaction, called the SMARTDOC viewer. This viewer (~6 MB) will be available for free download on the Web (compare with Adobe Acrobat Reader®). This viewer acts as a plug-in and will allow the user to view and examine the content of any SMARTDOC application component. This architecture enables distribution of small-sized application components, such as VolumeViewer, on the web. The SMARTDOC viewer connects directly to the window layout system of parent application, e.g. MS Word or MS Internet Explorer, and handles all visualization and interaction issues, including the communication with OpenGL. A SMARTDOC is an electronic document with embedded application components that will visualize the data described in the document. A user-determined view of the visualization will be used as the image when printing a SMARTDOC. A SMARTDOC will have the ability to set “bookmarks”, i.e. stored views that highlight specific regions of the dataset. These bookmarks are instant snapshots of the state of the application components at a certain time and includes various attribute settings and display properties. The bookmarks will be stored as part of the document and will be available the next time it is used. This feature can be useful when other users than the author use the SMARTDOC. Other users can use the document bookmarks and examine the dataset using the same visualization parameters as the author.

2.2. Visual User Interface

VolumeViewer has a Visual User Interface (VUI) as a complement to a traditional Graphical User Interface (GUI). The VUI enables the user to manipulate the graphical objects directly in the visualization with an immediate graphical response. Hence, the VUI improves the appearance of the application component and gives the user a more active role in the visualization. The interface for the user is the visualization of the data and the data itself. Examples of supported manipulating features are 3D orthoplanes and cropping. VolumeViewer also has a GUI that provides additional functionalities like the ability to browse for files and change object specific parameters.

2.3. Collaborative environment

Teleradiology allows radiologists to exchange knowledge by transmitting images, reports and comments to each other, even though situated remotely at different clinics or even at home. Although collaborative systems providing real time
visualization and navigation exist in a broad range of usage areas, their appearance in the medical area is very limited. The complexity of available systems enhances the needs for a solid framework that provides a collaborative and educational function in order to optimize the understanding of a medical visualization.

Most available applications for medical imaging can handle 2D images but very few can handle 3D volume data in order to render 3D objects. Those who can are often running on expensive workstations and are very complex to use. There is, however, an area that uses 3D graphics on networks from which knowledge of how to reduce complexity could be gathered - The computer gaming industry. Many of the games that work on ordinary PCs use the Microsoft DirectPlay API, a special library within DirectX that provides classes for networking facilities. Many of the tasks that VolumeViewerNet handles are the same as the ones of a traditional computer game, i.e. setting up a session, sending messages, player (user) management etc. VolumeViewerNet uses DirectPlay for all the necessary collaborative tasks. This approach to system architecture supplies, for the first time, the infrastructure for affordable and collaborative medical image analysis facilities that can handle 3D graphics. It also provides more availability to any member of a clinical management team and allows close collaboration between clinicians to optimize image interpretation and treatment planning.

### 2.4. 3D annotations

Most of the commercial software applications for medical imaging provide the opportunity to annotate 2D images. VolumeViewer has this functionality, but offers annotations in the 3D view as well. 3D annotations is a challenging issue, since it is hard for the user to specify a position in three dimensions with a simple computer mouse as the tool. VolumeViewer introduces a new method of placing 3D annotations, letting the user move a small 3D object in the scene. The object, a box, can only be moved in the normal direction of its faces, which limits its degrees of freedom but at the same time makes the process more intuitive.

### 2.5. Component interoperation

Although component technology provides the possibility to connect components from different origins, very few cooperative project solutions have combined product parts from different companies to form a new product. To provide DICOM compatibility of VolumeViewer, in other words integrate VolumeViewer with medical standards, DICOM components from the project partner AETmed and graphics components from AVS OpenViz have been fused together into one application component. VolumeViewer is a unique solution that uses selected parts from the AETmed Dir Server DICOM Toolkit.
3. Tools

The three versions of VolumeViewer are created using Microsoft Visual Basic® 6 (VB) and the visualization component suite AVS OpenViz 2.0. The collaborative environment is created using the Microsoft DirectX API. This chapter contains brief presentations of these tools and how they have been used in the development of the VolumeViewer component. It also includes an introduction to DICOM, which is a standard for medical imaging.

3.1. Microsoft Visual Basic

Visual Basic 6 has been used as the programming tool to design the user interface and for embedding the low-level components into VolumeViewer. Microsoft released their first version of VB in 1991 with the explosive codename “Thunder”. It was the first visual developing tool from Microsoft and it was supposed to compete with the big languages C, C++ and Pascal. Although, the first versions were weak competitors, late versions proved to be very powerful and have become a widely used professional programming tool. It is based on the simple programming language “Basic” but has gone through several changes since it first was introduced.

VB is an interpreted language, which means that the program compiles the code into an intermediate language, which can be interpreted by special drivers installed on the system. The interpreters translate the intermediate language to executable commands that the system can perform. This is the main reason why VB is a slow language compared to for example C++, which is directly compiled into machine commands.

The power of VB is that user interfaces such as command buttons and windows do not have to be programmed by the developer. The desirable controls are provided by VB and it is therefore very good for rapid prototyping (figure 2).

![Figure 2: User interfaces are easily created with VB](image)

It is very easy to use third party add-ons and COM (Component Object Model) object in VB. VolumeViewer makes use of the AVS OpenViz 2.0 class library to present and render the graphics. VB is also a tool for creating COM objects such as ActiveX components. COM is a software architecture that allows applications to be built from binary software components. ActiveX components are COM components that can be embedded into documents.
3.2. AVS OpenViz

AVS OpenViz is a low-level components graphics library that is used as the component basis for visualization programming in SMARTDOC. OpenViz provides low-level atomic components that mainly handle 2D and 3D interactions and visualizations of business data. However, there are some support for visualizing other data such as volumes and geographical data. The OpenViz Viewer renders all graphics in VolumeViewer and acts as the SMARTDOC viewer in this prototype.

OpenViz applications run on computers that have an OpenViz viewer installed. This viewer can be downloaded for free. Distribution of small-sized application components, which incorporates OpenViz, is feasible since the viewer only has to be downloaded once and contains the necessary functionalities to render OpenViz graphics. The application components are typically 200 kB.

3.2.1. Component Hierarchy

The concept of atomic components, functional components and application components can be used when working with OpenViz. OpenViz provides the low-level components layer, which typically are task specific high performance data structures. A characteristic atomic component is for example an OpenViz axis component. The user creates functional components by connecting several atomic components. Combining three axis components to form a 3D axis system can for example create a functional component. The 3D axis system can be used in combination with other functional components and a user interface to form an application component (figure 3).

![Component Hierarchy Diagram](image)

Figure 3: The component hierarchy

3.2.2. Scene Tree Architecture

The scene tree is a diagram of all components that explains how they are linked to the OpenViz viewer. The viewer is the OpenViz rendering window.

The scene tree consists of several nodes, which can be referred to as parents or children. A child can only have one parent while the parent can have many children. The nodes in the scene tree can either be geometry scene nodes or group scene nodes. Group scene nodes form the scene tree, while the geometry scene nodes contain the geometry that will be rendered.
There are OpenViz components that form the scene tree and hold geometry, but there are also components that handle the data flow. All data in the OpenViz pipeline is represented as an internal data structure called field. The field structure provides a single unified data representation for all data types. To render a dataset with OpenViz, it must be represented as a field. Most components have fields as both input and output. There are, of course, mapping components that read data and convert it into fields. Figure 4 illustrates the connection between the scene tree and the data flow. The input volume data is mapped into a field structure that can be read by the isosurface component. It is not uncommon that the input data is linked to several components as illustrated in the figure. The workbox component is a box that defines a region in space that its children should use. The domain component has a coordinate space large enough to accommodate all the data extents of its children. The domain maps all geometry node coordinates to fit into the space defined by the workbox.

3.2.3. Attributes
Except for the geometry nodes and the group scene nodes, the OpenViz scene tree also consists of attributes. An attribute is a parameter that specifies how a geometry should be rendered, for example surface opacity, line color or font size. Attributes are inherited from parent to children unless an attribute is set explicitly. When the scene tree is traversed all geometry will have the attributes defined in the parent root, if no other attributes are set.

3.2.4. Interactivity
The OpenViz tools offer a mixture of components that will help the user to build interactive applications. These components are divided into manipulators and interactors. A manipulator is a geometry (e.g. a plane) that takes user input to affect the rendered geometry, for example scrollbars or orthoplanes. An interactor is an object that processes events that are sent to the viewer. The
transforminteractor is one example. It catches the mouse events of the viewer and makes it possible to navigate in a scene.

3.3. Microsoft DirectX

DirectX is a Microsoft API (Application Programming Interface) collection. It is created to support game programming and consists of four main components: DirectX Graphics to use for graphics programming, DirectX Audio for audio programming, DirectInput to use for supporting input devices and DirectPlay to program multiplayer network games. The VolumeViewerNet collaborative environment is created on the DirectPlay API.

DirectPlay consists of a set of tools to be used for developing multiplayer network games. It provides classes that handle low-level network communication. The DirectPlay layer separates the network layer from the application layer, i.e. the application itself does not have to communicate with the network (figure 5). The application programmer has to make sure that the correct information and parameters are sent to the DirectPlay layer by specifying the addresses of the other participants that are supposed to receive the information. The API handles peer-to-peer sessions as well as client/server sessions and also enables voice communication with the special classes located in the DirectVoice class library.

![Figure 5: The DirectPlay layer separates the application layer and the network layer.](image)

3.3.1. DirectPlay Peer-to-Peer

The collaborative environment included in VolumeViewerNet is created with the DirectX peer-to-peer classes in the DirectPlay API. In a peer-to-peer session all involved computers are communicating directly with each other instead of through a server, which is the situation for a client/server session (figure 6 and figure 7). If a user wants to send a message to all of the other participants, the message must be sent as many times as there are users. It is obvious that the technique has scalability problems when the number of participants is large. However, because of the direct communication and the use of small messages everything that is needed to run a collaborative session is part of the user’s local application.
DirectPlay is media independent, which means that sessions can be held no matter what types of networks are involved. It supports the service providers TCP/IP and IPX on modem, serial and LAN connections. When DirectPlay is instantiated a virtual network connection is created between the application and the network which enables the application to always communicate in the same way regardless of which type of network that is used. When the virtual connection is created each user is assigned a unique address ID.

When parameters from the application are sent to DirectPlay, the API forwards the information to the network protocol. The information that is passed to the DirectPlay Layer is referred to as a message. The network protocol creates a packet, i.e. adds a number of bits to the message (for example physical destination address) and sends it forward. There is no restriction of the message size, however a packet cannot exceed the MTU (Maximum Transmission Unit) in size [1][18]. If a message is too large DirectPlay automatically fragments the message into smaller parts before sending it to the network layer. When a fragmented message is received, DirectPlay defragments it.

In a peer-to-peer session one of the participants must start the session and be responsible for the session. This participant is called the host. There is a possibility that the host has to leave the session early while there are still other users connected. DirectPlay offers two solutions, session termination and host
migration. Host migration is simply when DirectPlay migrates the host, i.e. makes another user the host.

3.4. DICOM

Digital Imaging and Communications in Medicine (DICOM) is a standard method for transferring images and associated information between different medical devices manufactured by various vendors [12][14][16]. It was initially introduced by the American College of Radiology and the National Electrical Manufacturers Association for communication (NEMA). DICOM is not an image format, but a set of protocols that an application, which claims to follow the standard, should conform to. One purpose of DICOM is to facilitate communication of medical digital images between different clinics etc. The standard is also meant for Picture Archiving and Communication Systems (PACS), which often have an interface to other parts of a hospital.

3.4.1. DICOM architecture

DICOM is built on an object-oriented architecture. There are two main types of components in DICOM, the Information Objects (IO), which can be seen in figure 8, and the Service Classes (SC). The information object contains the actual content, e.g. the image data and the patient name, while the service class specifies what can be done with these objects. There is for example a storage service class that enables devices to store the file. When an IO is combined with a SC they form a Service-Object Pair class (SOP). One common combination would be a CT Information Object and a storage service class, which together would form a CT image storage SOP class. This class would represent a storable CT study.

Vendors must clearly specify in their conformance statement what the role of the product is. In DICOM a machine can either be a Service Class User (SCU) or a Service Class Provider (SCP), which can be compared with the concepts of client (SCU) / server (SCP). This means for example that a CT scanner that claims to be a SCU of CT image storage class should be able to send images to a PACS that claims to be a SCP of a CT image storage class. If the two devices, on the other hand, do not conform to the same service they will not be able to communicate with each other even though they both follow the DICOM standard.
Figure 8: An example of an Information Object definition for a DICOM file
4. User requirements

Professor Alan Jackson at the University of Manchester, Division of Imaging Science & Biomedical Engineering, gathered the user requirements for VolumeViewer. A number of interviews with clinicians were conducted to guarantee that the application would agree with the needs of a possible end user. A questionnaire was given to 20 persons of whom eight are clinicians (neurosurgeons etc), five are junior radiologists, four are medical imaging physicists and three are basic scientists with extensive experience in medical image processing. The results from the questionnaire led to a list of functionalities that the application (VolumeViewer) should supply. The list of desirable features that all of the interviewees agreed on is:

1. The application should read DICOM native format data.
2. The application should be able to read data from any local directory to allow integration with standard PACS software workstations.
3. The application should provide the ability to identify a subset of the 3D imaging data to be included in the visualization.
4. The application should support variable isosurface rendering and volume rendering with controls over the isosurface, alpha and transparency values.
5. The application should allow three-dimensional interactivity with the rendered object including translation, rotation and magnification.
6. The application should allow controls over colour tables and if possible the lighting effects.
7. The application should support the placement of anatomical labels preferably with arrows to indicate the point of interest.
8. The application should allow the examination of the source data in any of the orthogonal planes with controls over window width and level.
9. The application should allow the display of personal data sufficient to identify the patient and should be configured so that all personal data can optionally be removed if data is to be exported outside a clinically secure environment.

There is also a list of features that one or more of the interviewees suggested would be desirable. These are the following:

1. The ability to upload annotated comments into the DICOM file. (6 interviewees).
2. The ability to simultaneously visualise multiple objects from matched data sets (e.g. bone from a CT data set combined with blood vessels from MRI dataset, two interviewees).
3. The ability to define multiple preset views with standardised orientation transparency and isosurface or volume rendering criteria for specific applications (two interviewees).
4. The ability to record combinations of visualization settings and to link these to hypertext links in the written report. This would allow the report to be linked to specific visualizations generated during the reporting process, which the clinician reviewing the report could then call back. (Two interviewees).

5. The ability to export individual images in a standard format such as JPEG or BMP (two interviewees).

6. The ability to export animations in a standard format such as AVI or MPEG, (one interviewee).

These lists have been the basis for the implementation phase of the study. Several features, other than the listed ones, have been implemented as well. Dr. Jackson has specified some of these and others come from other research [10].
5. Implementation

This chapter describes the implementation issues and the technical aspects of the VolumeViewer prototype in terms of the user interface, the visualization components and read/write components. The implementation conforms to the component thinking exploited in the SMARTDOC EC project (figure 9).

During the prototype phase the technical features of the application component has been prioritized. The graphical user interface of the prototype is presented but not thoroughly discussed. However, the accomplishment of a high-quality VUI, directly attached to the rendered graphics, is presented. The technical characteristics of the graphical features are presented as well as the read and write possibilities of VolumeViewer. The How-To and GUI information can be found in the VolumeViewer manual in appendix A.

The main focus of the study has been to implement a prototype component that meets the requirements of the medical end users and the objectives of the SMARTDOC project. VolumeViewer is because of this most suitable for medical datasets, but it can be used for other types of volumetric datasets as well, including geological and meteorological datasets.

As mentioned earlier, VolumeViewer is developed in three different versions to cover the usage areas:

1. Study and diagnosis (VolumeViewer)
2. Collaborative analysis (VolumeViewerNet)
3. Presentation and journaling (VolumeViewerX)

The first version is a stand-alone application, which can be compared to many of the existing software packages in the area. The second is an expansion of the first to also allow collaborative sessions over a network. The last is an ActiveX version that can be inserted into web documents as well as Microsoft Word or PowerPoint documents.
5.1. User Interface

The importance of good application user interfaces has been investigated and evaluated in a number of research projects. Often, a good GUI is in the center of these evaluations. However, a user centered VUI is the core of the application component presented in this report. Although not the focus in this study, the GUI must not be underestimated, since it is utterly important when creating a user-friendly application [10]. Basically, there is no right or wrong, since the user interface depends of the functionality of the application. Building a good user interface was not the issue for this phase of the project. However, some “rules” of making interfaces and some new ideas have been used. For a thorough presentation of the user interface of VolumeViewer, see appendix A.

In VolumeViewer the volume data is represented as 2D slices and a 3D volume in two 2D views and one 3D view (figure 10). This layout was developed by the authors, but can also be found in other systems. To interact with the volume data, VolumeViewer makes use of GUls built in VB and VUIs built from OpenViz components. The GUI consists of a set of controls, for example sliders, buttons and checkboxes, which the user can use to change a variety of parameters. The controls of the GUI are appropriately grouped together for easy and intuitive access. For example, all isosurface parameters appear together, surrounded by a frame.

In a VUI, the user is given an active role in the visualization. It consists of 3D objects that are integrated in the rendered scene. These objects give an immediate graphics response when they are affected. A VUI limits the needs for additional windows with menus for different parameter settings. The ability to move orthoplanes through the volume dataset, the crop function, panning, zooming and rotation are all examples of interaction that is performed through a VUI.

Figure 10: The GUI and VUI of VolumeViewer.
5.1.1. Orthoplanes
The orthoslices, described in 5.2.2, in the 3D view are combined with visual components called orthoplanes. The three orthoplanes (one in each coordinate plane) are important parts of the VUI of VolumeViewer. It provides the ability to determine the location of the orthoslices by dragging them along their normal axis direction directly in the visualization. The location of the orthoslices is synchronized with the location of the orthoplanes. The orthoplanes are equipped with eight “handles” each that make them easy to grab and move.

5.1.2. Cropping
The datasets used in medical imaging are often very large. Even on fast workstations it can be very time consuming to render the whole dataset at all time. It is often the case that the user is only interested in a small part of the volume and therefore it makes sense to crop the dataset before the actual examination takes place. VolumeViewer supports cropping in the 3D view.

Cropping is done through the VUI. Six scalable planes form an interactive 3D bounding box, which manipulates the extents of the volume dataset (figure 11). The user resizes the box and the dataset is immediately cropped according to the extents of the box. Since the crop method gives an immediate feedback on the user’s action, the method is very easy to use.

It is also possible to remove the isosurface totally if the user is only interested in viewing the 2D projections. This will of course further increase the speed of the rendering process.

![Figure 11: The cropping box. Each plane can be moved in the normal direction of its face. The selected plane is highlighted in yellow.](image)

5.1.3. 3D Light
VolumeViewer offers the user to change the position of a directional light source in the 3D view. This can be useful if parts of the isosurface obstruct other parts. The directional light can be switched on and off. A default static directional light will be used if it is switched off.
The light source is represented as a small pentahedron, which can be moved independently from the volume (figure 12). The light is positioned on the surface of a sphere surrounding the workbox.

5.1.4. Orientation indicator

In many cases it can be hard, even for experienced users, to know how the volume is oriented in space. Even though medical datasets have a natural correlation to the human body it can be difficult to comprehend what is actually displayed, especially if the dataset has been cropped.

VolumeViewer has an orientation indicator that always rotates with the volume and thereby helps the user to know what is up and down etc (figure 13). The arrowhead is color coded in four different colors to further increase the understanding of the orientation.

5.1.5. Resizing of the application component window

VolumeViewer can be resized to provide the user as big work area as possible. The obvious solution is to check the window size immediately after a resize event. If the size is above or below the allowed values, the window auto-resizes to the closest allowed value. To avoid that the window first is resized to a non-allowed value then resized back to the closest allowed value, another method involving sub classing is used.

Sub classing

The Microsoft Windows® operating system uses messages to communicate between windows (including buttons and other controls) and the operating system. A message is basically a unique value that tells the receiver about an action that has to be executed or about an action that has taken place.

Each window has a message handler that evaluates the message and performs the appropriate action. A window message handler is called “WindowProc”. When sub classing is used, a new message handler is created and inserted in line with
the original message handler. The message sent to the window is handled in the
new message handler before it is sent to the original handler (figure 14).

![Diagram](image.png)

Figure 14: The Sub classing loop

In VolumeViewer the application window is subclassed. When the resize window
event is called, a virtual VolumeViewer message handler is called. Its task is to
ensure that it is impossible to enlarge or make the application window smaller
than the allowed values. When manipulating the Windows message chain, it is
important to make sure that the new message handlers are disabled and pulled out
of the chain when they are not needed.

5.2. Visualization components

The volumetric datasets that are loaded into VolumeViewer are displayed using
special visualization components that each performs a specific task. The
visualization components are of course of most importance, since they are
responsible for the actual presentation of the data. VolumeViewer supports
isosurfaces, orthoslices, color mapping and annotations in 2D and 3D.

5.2.1. Isosurface

Isosurfacing is a very fast and effective way to visualize volumetric data. An
isosurface is a surface in space where some function is constant. In this case the
function is the scalar value at each vertex in the field. In medical imaging,
isosurfacing is for example used to examine bony and vascular structures to locate
and identify complex 3D compositions prior to surgery or radiological
intervention. Facial reconstruction is an area that particularly uses this technique.
The algorithm used in OpenViz to create isosurfaces is “Marching Cubes”, which
in short terms can be described as follows [2][11]:

Each voxel has eight vertices with a scalar value associated to it. The algorithm
examines each voxel by “marching” around the structured grid. The idea is to first
find the voxels that are intersected by the scalar value (isovalue), i.e. voxels that
have at least one vertex with a larger value than the isovalue and at least one with
a smaller value. The voxels whose vertices are entirely outside or inside the
surface are not interesting at this stage. In a second pass through the selected
voxels, triangles are created according to a specific set of rules (figure 15). An
assumption is made that each edge is intersected by the surface at most one time.
This leads to a total number of 254 ways that the voxel can be intersected. If
rotation over a primary axis, inverting the state of the vertices or mirroring the
shape across any of the primary axis is not counted for it reduces to 15 ways. The
different cases are saved in a lookup table, which is never changed. The position
of the surface’s intersection with the voxel is calculated using trilinear interpolation. Finally the triangles can be rendered.

Figure 15: Examples of the possible voxel combinations. The blue corners have been tested to be inside the surface. The green triangles will be part of the resulting isosurface.

The user can change the isovalue, color and opacity of the isosurface in VolumeViewer.

5.2.2. Orthoslices

VolumeViewer implements orthoslices, which allows the user to view a 2D slice through the volume. They are synchronized with the orthoplanes described in 5.1.1. This can be valuable for medical doctors who are used to examining a volumetric dataset by viewing the 2D images produced by, for example, a CT scanner. Another reason for presenting the data in 2D, as a complement to the 3D view, is that 3D volumes created from CT scanners and MRI are created artificially. A 3D volume dataset is created when all 2D slices are merged together. This process may generate artifacts. Details in a very complex dataset may also be more detectable in a 2D view.

In VolumeViewer the orthoslices are visually represented in the 3D view as planes through the volume. They can be moved manually through the volume through the VUI. The user uses special handles provided by the VUI to select the orthoslice in the normal direction. Each plane is orthogonal to one of the axes in the scene. The application also lets the user view the orthoslices in separate orthographic views to avoid possible artifacts that may arise when perspective projection is used in the 3D view.

The user can change the opacity of the orthoslices. This operation only affects the orthoslices and not the opacity of the 2D views. VolumeViewer lets the user decide which of the orthoslices should be visible in the 3D view. It is also possible to change the color map of the orthoslices.

Color mapping

One important feature supported by VolumeViewer is color mapping of the data. This means that the scalar values in the structured field are mapped to specified colors. The most commonly used color map is the grayscale map where the highest value(s) are mapped to white and the lowest to black. Sometimes other maps than the grayscale are better suited. In VolumeViewer the user can choose from two predefined maps (grayscale and hot metal) or define a new color map by specifying the two boundary colors. The chosen colormap will affect the orthoslices in the 3D view as well as the 2D views.

It is also possible to change the level for the maximum and minimum values. Medical images, for example, often have 4096 different intensity levels. If a grayscale colormap is used the intensity 4096 would be mapped to white. By changing the maximum value to 4080 all the values ranging from 4080 to 4096 would be mapped to white. The remaining intensities will now be mapped to a
greater span of gray levels and since the human eye can only distinguish quite a small number of shades of gray levels, this will improve the readability of the image. It is also possible to increase the minimum value so that more low intensity levels will be mapped to black. Figure 16 shows a CT scanned slice through a skull viewed in VolumeViewer. The left image shows the original image with a span of 0-4096, while the right image has decreased the span to 1760-3592.

![Figure 16: The left image has a min value = 0, max value = 4096. The right image has a min value = 1760, max value = 3592.](image)

The right image has mapped more intensity levels to totally black and totally white respectively, while the intensities in-between has a wider range of gray levels to be mapped to.

### 5.2.3. Annotations

It is very important for the user to be able to annotate interesting structures in the dataset. VolumeViewer, as mentioned earlier, supports annotations in both the 3D view and the 2D views while most commercial applications only support annotations in 2D.

#### 3D annotations

3D annotations is a problematic issue, since it is hard to find an easy and intuitive way to position them in the 3D view. VolumeViewer uses the already existing orthoslices for this task. The user must first set VolumeViewer in “annotation mode” and then position a small box at the point where the annotation arrow will start (figure 17). The box is always positioned at the intersection between the three orthoplanes and hence it is moved whenever one of these is moved. When the box has been placed at the desired location the user specifies the associated text. Once the annotation is confirmed the green box will disappear.
Figure 17: Example of a 3D annotation. The green box indicates the position, at which the annotation will be pointing. It will not be visible once the position is confirmed.

**2D annotations**

The user can draw an arbitrary shape in the 2D views that will be part of the 2D annotation (figure 18). A 2D annotation will not be visible in the 3D view and will only be visible in a 2D view when the plane, in which the annotation was created, is visible.

![2D Annotation](image)

Figure 18: Example of a 2D annotation

The text of both 2D- and the 3D annotations can be modified at any time. It is also possible to change the colour of the annotation text and the annotation arrow.

**5.2.4. VolumeViewer scene tree**

VolumeViewer uses many components provided by the OpenViz library. The scene tree is huge, but it can be interesting to see an overview of some limited parts of it (figure 19). Notice how the orthoslices are used in several domains, which corresponds to their appearance in multiple views (2D and 3D) in VolumeViewer.
5.3. Read and write components

VolumeViewer supports AVS volume field files, binary RAW files and DICOM files as input. The AVS volume field format is an internal format for AVS products. This format has been used for testing purposes. These files are read using the OpenViz ReadVolume component. VolumeViewer can store parameters in external binary files associated with the volume files. VolumeViewer automatically searches for such files when a volume is loaded into the application component.

5.3.1. Raw data reader

Raw data is a primitive binary data format that is used to represent the volume data information. Most files only include the pixel values of the dataset, but it is not uncommon that they contain the dimensions of the dataset. In both cases VolumeViewer extracts the pixel information and stores it in a 3D array. This array is passed to the OpenViz component StructuredFieldBuilder\(^1\), which maps the data into an OpenViz field. Represented as a field, the data can be passed to other components that visualize it.

5.3.2. DICOM reader

Since VolumeViewer primarily has been designed to be used for medical purposes it is of great importance that it can browse and load DICOM-files. One of the partners of the SMARTDOC consortium, AETmed, is specialised in advanced information and communication technologies for medical imaging applications. The open component hierarchy of VolumeViewer makes it possible to interface one of their products, Dir Server DICOM, and can therefore browse DICOM databases and load DICOM files.

\(^1\) A Structured Field is an OpenViz field, with the data information ordered in a structured mesh (uniform, rectilinear or curvilinear). Each data value is located on a mesh point.
In practice VolumeViewer calls a DLL that will display a GUI for browsing the DICOM database. This GUI is a small part of AETmed’s software DICOMed Review. Two temporary files will be created when the user has chosen a study of a patient, one RAW file and one INI file. The RAW file contains the binary image data (pixel data), which can be read by the raw data reader. The INI file contains some of the attributes from the DICOM Information Object definition (figure 8). The attributes are modality, patient ID, patient name, referring physician's name, study ID, study date, slice thickness and spacing between slices. An INI file reader reads the INI file. The information stored in the INI file can be displayed for the user at any time.

5.3.3. Storing of parameters
One key feature of the SMARTDOC concept is the possibility to store parameters such as views of the 3D graphics, attributes and annotations. When VolumeViewer is terminated these parameters are stored in an application-specific binary file. When a volume data file is loaded into a new VolumeViewer session, it searches the dataset directory for a file associated with the volume data file and reads the available information.

A view is the direction from which the volume is being observed and possible zooming and panning parameters. A view includes not only the transformation matrix of the isosurface, but also its attributes, e.g. the opacity and the isosurface level.

5.4. ActiveX implementation
The third and last version of VolumeViewer is the ActiveX component. ActiveX components can be inserted in all documents that allow embedded controls, such as Microsoft Word, Microsoft PowerPoint and Internet Explorer. ActiveX is not a programming language, but a set of technologies and a set of rules of how different applications should share information. ActiveX is built on COM (Component Object Model) technology. COM is a low-level specification by Microsoft on how components should be integrated and also how various components should interact and share information. The benefit of COM is the ability to reuse previously existing components, which clearly is a time saving and efficient way to build larger systems. VolumeViewerX has been developed as an ActiveX control, which simply is a control that incorporates ActiveX technologies. ActiveX controls can, in many ways, be compared to JAVA applets, but are much more powerful since they have access to the Windows operating system.

VolumeViewerX had to be redesigned in order to be more suitable for the document environment. One limitation for the component is its width. When a VolumeViewerX component is to be inserted in a Microsoft Word document it should not have a width of more than 210 mm (the width of an ordinary A4 page). The GUI has been moved to a separate window to give the 2D and 3D graphics as much space as possible, since these are the most interesting parts of the

---

2 A DLL (Dynamic Link Library) is a set of small service providing programs that can be used by larger programs.

3 A Windows initialising file.
application. All the functionalities of VolumeViewer can be found in VolumeViewerX as well.

5.5. Collaborative Environment
The sharing of visualization objects over large distances can be referred to as collaborative visualization [6]. The functionality and design of collaborative visualization environments are depending on many different factors, for example what parameters to share, location of data, available bandwidth and process distribution. One common solution is to divide the participants into one master and one or many slaves and transmit images of the master screen to the slaves. This is for example how the Microsoft NetMeeting application works. Other solutions are based on a shared event model, where an event is generated each time a parameter on a host application is changed. The event can be multicasted to other participating applications without sending any pixel values. All parameters are then properly defined at each user’s application unlike the master and slave solution. The collaborative solution suggested in this report is based on the share event model, having the actual dataset stored locally.

SMARTDOC introduces “collaboratories”, application components including collaborative environments for advanced real-time 3D data interaction over remote geographical distances. VolumeViewerNet is the collaboratory created on the VolumeViewer foundation and the DirectPlay class library. Current research in the area of collaborative visualization includes some web enabled studies [5][7] and others with more sophisticated equipment like super computers [3]. The VolumeViewerNet collaboratory has the same functions as the stand-alone application, but also includes an extra dialog box with some additional functions necessary for networking. The only communication between the two users in a session, except for the interaction with the volume, is done through a traditional text chat. Datasets are distributed separately before a session and stored locally.

5.5.1. Architecture
The overall architecture of a collaboratory is based on the DirectPlay network abstraction layer and a collaborative framework provided by the SMARTDOC EC project. By using the network abstraction layer VolumeViewerNet shields the participating users from the underlying complexities of varied connectivity implementations and enables them to focus on the real-time navigation scenario. The collaborative environment created from DirectPlay supports the session management process and enables the user to join, create, leave or search for available sessions. The collaborative framework provides state and event synchronization, which means that the applications of all participating users are immediately synchronized with each modification of a state, or the occurrence of an event on a host application. When a session is initialized each user’s user interface is synchronized with that of the host application.

A collaboratory consists of a viewer component, which contains the visualization. It is responsible for rendering the dataset with consideration to the user-operated parameters and all input mouse interaction. Thus, the viewer has data input in the terms of saved parameters and the dataset itself, and input from user interaction. The communication object administrates a collaborative session. The viewer gets
additional input from the communication object, which receives event messages. Figure 20 illustrates the architecture of two collaboratories.

![Collaboratory Diagram]

Figure 20: The overall architecture of a collaboratory

The collaborative environment currently supports peer-to-peer sessions. The application implements the DirectX classes to create a DirectX object and a DirectPlay object that can open a peer-to-peer session. A DirectPlay event object is also created. This object captures all events engendered by the peer object, such as the createPlayer event when a new participant joins a session, or the destroyPlayer event when a participant decides to leave the session.

When a user wants to join or create a collaborative session, the createPlayer event is called and a DirectPlay player is created. The reason to call it “players”, instead of for example participants, is of course depending on the library’s original usage area – network gaming. The creation of a player enables the ability to send and receive messages. All communication takes place between DirectPlay players, not between different computers. In the VolumeViewerNet collaborative environment the players are, more appropriate, called members.

When connecting, i.e. joining or creating a session, the collaborative environment automatically lists the different service providers on that particular machine, which is a great advantage since VolumeViewerNet is supposed to work on both desktops as well as laptops. The user that creates a session is creating a direct communication channel between all involved computers. This user is called the host. Although the basic technology is peer-to-peer, the collaborative environment implements some client/server thinking.

5.5.2. Active and Passive Members – Collaborative Rules

The two members in a VolumeViewer collaborative session are defined to be either active or passive members. This division is due to the concept of the collaborative rules, which defines the state/event synchronization process. The collaborative rules are defined as follows:
1. Only one user is able to perform operations with the application. This includes all interaction and all parameter settings. This user is recognized as the active member.

2. The host is the only one that can distribute or reclaim the active membership.

3. The user that does not have the active membership is known as the passive member. The passive member has no possibility to affect the session. The only actions it can perform is to use the chat, request the active membership or disconnect.

These rules state that the passive member is just a spectator watching the active member interacting with the application. All interaction on the active member side is briefly stored and sent to the passive member. The information sent is received at the passive member side and the two application components are synchronized. The host, the user that started the session, has the ability to decide when the other user will have the active membership and can always reclaim it. The passive member can alert the host when he/she wants the active membership. When a user is a passive member all the features but the chat, the disconnect function and the request active membership function are locked. If the active membership is received all features are unlocked.

The reason for the collaborative rules and the division into active and passive members is to prevent multiple users affecting the volume data at the same time.

5.5.3. Messages

When the active user performs an action that changes the state of the application or executes an event, the parameters that were modified will be sent to the passive member embedded in a DirectPlay message. All such messages consist of a buffer that holds the data information. The first thing added to the buffer is the message type. VolumeViewerNet uses several message types that will be discussed further on. The actual information can be divided into data or strings. Strings cannot be sent as data. This division exaggerates the needs for the message type information. The receiving application must know in what order data and strings come in order to extract all information correctly. A typical buffer is illustrated in figure 21.

![Figure 21: A typical DirectPlay buffer](image)

The message will be part of a network packet that will be received by the DirectX layer at the receiver. When the message is retrieved, DirectPlay will transfer the message to the application that places it in a message receive queue (figure 22). As already mentioned, a message is defined by its identifier, the message type, which is the first data information in each message (figure 21). The passive
VolumeViewerNet uses a number of different messages, which need to be considered when they are received:

- The matrix message: When the volume is affected by translation, rotation, zooming or panning the matrix messages will be sent to the passive member. To avoid a massive flow of matrix information, this message is only sent when the action is performed, i.e. when for example a rotation is made.

- The isosurface message: When any of the isosurface parameters (isovalue, surface opacity, surface color and visibility) is changed this message is sent.

- The orthoplane messages: When any of the orthoplane parameters (location in the dataset, opacity and visibility) is changed this message is sent. The color map information (color map colors, window level and contrast are sent in a separate message).

- The file message: This message is the most important one in the beginning of a session. It is sent by the host to a joining application and contains all information of the host’s dataset that is necessary to determine if the “joining” dataset is the same. The dimensions (x,y,z) and the filename are compared. The fileOK message is sent by the joining application when the comparison between the datasets is done. If the datasets do not match the joining application will automatically disconnect. When the host application receives the fileOK message it either adds the member to the current member list or erases the information about this member.

- The 2D and 3D annotation messages: These messages are very important during a collaborative session. Except for the chat this is
currently the best way of communicating with other users about the datasets. The 3D annotation messages hold x,y,z values and text information. Several annotations can be sent in one message. The 2D annotation message is more sophisticated. It obviously contains the text information, but it also holds a number of parameters and data in order to represent the associated 2D polyline. These parameters include polyline data representation, in which plane it appears and the number of points. If more than one annotation is sent the start and stop values for the polyline have to be considered as well.

There are additional messages that for example hold chat information, viewport information and advanced options parameters. All messages are built the same way using buffers as described earlier.

5.5.4. Information Protection

When a user joins or creates a session it cannot be guaranteed that the information and parameters about that particular dataset will be stored, since the dataset parameters are updated according to the active member’s interaction. The user has one possibility – to save views in the advanced options dialog box. However, this action does not save all information. It is important to be aware of this when joining or creating a session.

Annotations, which are considered to be the most important information, are treated with extra care during a collaborative session. The annotations made by the active and passive members are given a unique identity to avoid name conflicts. After the session, all annotations are merged together and the user can save the annotations of importance. While the session is running the user can only edit or delete his/her own annotations.

The integrated external DICOM component provides professional network security for medical data allowing VolumeViewer to focus on the visualization and collaboration features of the system [13]. Local client-side volume image data is anonymized (figure 22).
6. Comparison
During the development of the three versions of VolumeViewer, other software packages and collaborative techniques have been evaluated in order to get an understanding of how data can be displayed and to get new ideas on how to implement different functions. Three software packages in the area of 3D volume visualization are assessed and compared with VolumeViewer. Collaborative research is getting more and more common and thus, there is an increasing amount of collaborative solutions available. Two collaborative solutions are described, which both have similarities to, and differences from, the collaborative implemented in VolumeViewerNet.

6.1. Application comparison
The VolumeViewer is compared to the demo versions of three commercial 3D packages (table 1). These systems are Amira, 3D Doctor and ActiViz COM. Amira is a more general visualization software package, while 3D Doctor is a medical imaging application. ActiViz COM is an interface to the Kitware VTK library of visualization classes.

6.1.1. Amira 2.3
Amira is an advanced high performance visual programming software that supports a number of file formats. The user can use a large set of different modules and connect them in a preferred way to create visualizations of a dataset. It has a wide range of functions and visualization methods that a user might want to use. It consists of three different windows: a viewer, a visual programming window and a command prompt. Due to the usage of many windows and a poorly designed GUI Amira is not very user-friendly. It requires deep knowledge in many visualization techniques to be able to use the application properly. However, Amira is a high performance application that includes an impressive spectrum of implemented functions.

The volume dataset is displayed in one or several 3D views. There is no specific 2D view and no use of VUIs. Rotation, zooming, panning and translation can be done directly in the 3D views, but all other parameters must be defined in the GUI. An example that demonstrates the advantage of a VUI becomes evident when the location of an orthoslice must be defined by a slider in the GUI, instead of having the possibility to drag the actual slice in the view where it is displayed.

Some functions are very hard to use. An example of this is the placing of 3D annotations. The 3D annotations are placed in the volume data, which is a very problematical operation. The annotation might seem to be placed correctly from one point of view when it is in fact not in the right position. It is a well-known problem to position something projected on a 2D screen in a 3D world.

After installation, the size of the Amira folder is approximately 50 MB. This can be compared to the size of VolumeViewer, which is approximately 200 kB, plus the 7 MB of the SMARTDOC viewer.
6.1.2. 3D Doctor

3D Doctor is an application for medical 2D imaging, but allows surface rendering and primitive volume rendering. The program can import many file formats but has a limited set of functions. This package also uses multiple windows that can be troublesome to control. The application makes no use of VUIs and the GUI is limited to a menu list and a menu bar with a number of command buttons.

The 2D slices of a volume dataset are the most important elements when working with 3D Doctor. Thumbnails of all slices are displayed in a separate window. Individual slices are selected in this window and are enlarged in an additional window. To create surface renderings or volume renderings the user has to mark regions to be rendered in all slices. This can be done automatically or by hand.

3D Doctor only supports annotations in 2D, which are represented as plain text in the slice image. The annotations are only visible in the slice in which they were set. In general, the features of the software package are very limited. The size of the installed application is approximately 7 MB.

6.1.3. ActiViz COM

ActiViz COM is a Microsoft COM interface to VTK, Visualization ToolKit provided by KitWare. It is designed to let the user insert 3D visualizations into any document supporting embedded controls, e.g. Excel and Word. The VTK library includes more than 600 classes that the user can use to create visualizations.

The functionality of ActiViz is in many cases the same as in OpenViz. To use the libraries, the users have to be familiar with programming and visualization techniques. VolumeViewer, on the other hand, has been built to a higher level of abstraction, which isolates the user from programming.
### Table 1: A comparison of VolumeViewer and other similar software packages

<table>
<thead>
<tr>
<th>Software/Features</th>
<th>VolumeViewer</th>
<th>3D Doctor</th>
<th>Amira</th>
<th>ActiViz COM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICOM compatible</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>2D Annotations</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>3D Annotations</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Orthoslices</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Isosurfaces</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Isolines</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Volume Rendering</td>
<td>NO</td>
<td>YES, very primitive</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Cropping</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Color mapping</td>
<td>YES</td>
<td>?</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Store views</td>
<td>YES</td>
<td>YES, as a project</td>
<td>?</td>
<td>NO</td>
</tr>
<tr>
<td>Collaborative possibilities</td>
<td>YES</td>
<td>NO</td>
<td>?</td>
<td>NO</td>
</tr>
<tr>
<td>Insertable into documents</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

6.1.4. Discussion

Although VolumeViewer is a small application component of only 200 kB (not including the size of the viewer) and the analysed packages are commercial applications, it is a worthy competitor. The present version does not include functions like volume rendering or isolines as some of the other packages do. However, the use of VUI as a complement to the GUI is a good combination that makes the application component very easy to use.

6.2. Collaborative solution comparison

Two collaborative solutions, which provide 3D interaction, have been explored in order to get new ideas of collaborative work and to be able to understand and establish the advantages of VolumeViewer. Common subjects for collaborative visualization are for example molecular data [17] or meteorological data [5]. The collaborative solution created on the VolumeViewer foundation is compared to the MICE project (Molecular Interactive Collaborative Environment) and the NCSA (National Centre for Supercomputing Applications) Habanero collaborative framework [4][15].

6.2.1. MICE (Molecular Interactive Collaborative Environment)

MICE is a specialized tool for visualizing complex macromolecular structures. It is written in Java and uses VRML or Java3D to represent the molecular scenes. These scenes can be imported or be created as geometry within the application.
MICE works as a stand-alone application but the user can choose to share the viewer of the application with other users. Sharing a viewer means that remote MICE users can download the dataset and view the interaction taking place in it. The MICE application automatically locates ongoing MICE sessions that can be joined.

The collaborative solution uses client/server technology implemented in the ICE (Interactive Collaborative Environment) API, developed at San Diego Supercomputer Center. ICE handles the communication distribution and the synchronization process of the involved clients. A central server manages all state event communication during a session. The API manages user identification and ownership and includes essential tools for message ordering.

6.2.2. NCSA Habanero

The Java written Habanero environment enables multiple users to synchronize events and states on multiple copies of a software application. Users can use a set of available applications in order to perform collaborative interaction with datasets, or create their own applications by using the tools provided by the Habanero framework.

Like the collaboratories described in this report, the Habanero collaborative applications share all the important states and events. When an application joins a session, all necessary parameters and settings are sent to ensure that the different copies of the applications see exactly the same state. Habanero applications use arbitrators that define specific session rules. Arbitrators can for example ensure that only one person can perform actions. A lock is a special kind of arbitrator, which is used to disable certain functions of the application.

The environment supports session management and implements client/server technology. It enables the user to create, join, browse or leave sessions. The server hosts a session and the clients interact with the session using applications called “Hablets”. A strength of the environment is that the clients have record and replay possibilities. It is possible to join a session or view replays through a web browser using a special “helper” application. The environment also provides additional features like inviting users by emails or pop-up messages and anonymity.

6.2.3. Discussion

There are several solutions for collaborative environments and work over geographical distances. The two solutions presented in this chapter were chosen since they include distinct differences and similarities with the collaboratories defined in this report. They also demonstrate examples of usage areas, others than medical imaging, where collaborative work is used.

Unlike a collaboratory, the ICE API is designed for transmitting whole VRML represented volumetric datasets between the involved clients. This implementation demands high performance central servers and network channels to avoid scalability problems. If many users join a session at the same time the server must send the dataset to all users at the same time, which can be problematic. A collaboratory, on the other hand, only uses small state event messages during a session reducing the amount of network traffic as much as possible. The concept of sharing a viewer can be compared to VolumeViewerNet.
Since the controls of a passive user are disabled, a passive user’s application can be seen as a second screen, which is the case in the MICE project.

The Habanero collaborative environment is a framework that can be compared to DirectPlay. It provides classes for session management that can be used to create collaborative applications. In the Habanero case, these applications are called Hablets. In this report they are called collaboratories. Significant for both solutions is that the framework advocates the use of small state event messages. Both frameworks use some kind of collaborative rules. The rules in Habanero can be compared to the VolumeViewerNet collaborative rules. In Habanero, a special arbitrator has to be specified to ensure that only one person at a time has the permission to perform actions. This is always the case in the present version of VolumeViewerNet. The locks in Habanero can be compared to the passive membership in VolumeViewerNet. A passive member can never perform actions. The arbitrators and the locks in the Habanero solution are more sophisticated than the VolumeViewerNet collaborative rules. However, the collaboratories have a main focus on ease-of-use. The many options of how restricted different actions can be in Habanero can be cumbersome to specify and understand.
7. Assessment

The VolumeViewer application has been evaluated by the end user and a specialist in medical imaging to improve its usability and performance. Two diploma workers in the area of graphical user interfaces in medical imaging have also been consulted. Systematic discussions about the features and future possibilities have taken place several times, which have resulted in additional features that were not among the original user requirements. Examples of these features are for example the direction indicator. The end user’s general opinion of the application is that it is very useful. However, after the discussions it is clear that some work on the GUI needs to be done.

Most of the user requirements presented in chapter 4 have been implemented. Although the collaboration with AETmed has led to a fully functioning DICOM solution, which corresponds to some of the requirements, a more independent solution is crucial since VolumeViewer cannot require installation of additional software packages to work properly. Hopefully, future collaboration will lead to a more autonomous solution.

Volume rendering is desired but not yet implemented. This visualization technique is not yet fully developed or trusted to be used in the diagnosis process. It is more important for initial guiding in the medical work and for the ability to do nice and understandable presentations. However, with improved reliability it can contribute to the radiological work and will probably be implemented in VolumeViewer in the future. This will also enable the requirement to import more than one dataset, e.g. a CT bone dataset combined with blood vessels from an MRI dataset.

New suggested functions that might be implemented are:

- perform zooming in the 2D views.
- create color maps that span over three colors.
- hide two of the views in order to maximize the third view.
- display only those annotations that are selected. Annotations should also be highlighted or visible (if invisible) when the users click on them.
- Perform cropping by specifying the dimensions of the crop box in a dialogue window.
- The orientation indicator could be represented as a stylistic character instead of an arrow. This could improve the ability to distinguish for example front, back, top and bottom. Another solution is to add the initial letters of the words front, back etc. These letters could be visible in 3D view as well as in the orthographic views.

The box that is used when placing 3D annotations should be visible in the 2D views as well.

7.1.1. A customized ActiveX component

The end user fully agrees on the SMARTDOC concept with three different stages for diagnosis, collaborative analysis and presentation/journaling respectively. An
The idea to extend the presentation/journaling part would be to let the user create and publish reports, which includes custom VolumeViewer ActiveX components. These reports could be posted to a web server from which they could be easily accessed and downloaded (figure 23). Such a solution would have great potential according to the end user.

A custom made ActiveX component means that the component can only read and display a specific volume dataset defined by the creator. A customized ActiveX component would have the ability to change parameters as in the stand-alone application, but would not have the ability to create new annotations or store new views. The annotations and the views that were stored by the creator of the component would of course be visible. The component would only work when displaying that specific dataset. These customized components would not be chargeable since their use is very limited. The stand-alone application from which these customized ActiveX components would be created would however be. The customized components would require the SMARTDOC viewer which would be available for free. This is an excellent idea to spread the use of VolumeViewer.

7.1.2. Future assessment

The end user, Prof. Alan Jackson, is planning to put together a special group of medical students who will use the SMARTDOC VolumeViewer in order to test its usability in the general radiology work. The results of previous assessment sessions (described earlier) will first be carefully considered to improve the application component. This group will perform the first large scale test of the SMARTDOC concept.
8. Conclusion
This report has presented the new ideas and concepts of the SMARTDOC project including a detailed description of the development of a SMARTDOC VolumeViewer prototype. The prototype has been demonstrated for the consortium members as well as for several possible end users and they are all very enthusiastic about it. The need for functionalities that SMARTDOC components offer are widely expressed and pointed out in research articles [10].

It is totally clear that there are holes to fill in existing systems, especially in medical image management. There is a lack of intuitive, clever and convenient ways to handle data on ordinary desktops or laptops. SMARTDOC components would handle these problematic situations by offering a way to not only bring and present data on those machines, but also to provide the interaction needed.

The next step is to improve the VolumeViewer component to provide all the required features, including an enhanced and neat user interface. A more independent DICOM solution must also be implemented. A DICOM reader less connected to AETmed is a necessary step in order to guarantee the use of the component. The components must be independent, easy-to-use and as user-friendly as possible.

The next phase of the SMARTDOC project it is to create other applications in other usage areas employing the experience gathered during the VolumeViewer development process. It might seem strange to use such a low-status and high-level programming interface as VB when developing the application components, but it is wrong to underestimate the many advantages of the tool. Graphic applications are very dependent on speed, which eliminates VB as an ideal language in which to write a 3D graphics application. This is however overcome due to the fact that the OpenViz classes are written in C and C++. Future development will go on using the new independent .NET technology from Microsoft. It provides the ability to use an environment that allows rapid application development and is considered as the best option for fulfilling the project goals. The .NET technology was not officially released in time to facilitate the prototype development presented in this report.

When asking medical doctors about a user interface for medical imaging, the answers often correlate: “The application should only consist of the medical images and / or the volume”. However, developers know that some kind of user interface has to exist. In the SMARTDOC project, visual user interfaces are used as a complement to the graphical user interface. This seems to be an efficient and accepted way to meet those requirements. The comparison with other software packages in Chapter 6 provided knowledge of the functions that should be included in software for 3D visualization. None of the packages that provided the possibility to use annotations has intuitive or smart ways to position them. Hopefully, this has led to the fact that annotation placement in VolumeViewer is easy and intuitive. The comparison has also led to VolumeViewer not having multiple windows, except for VolumeViewerX that has all the controls in a separate window.

There are multiple ongoing or complete solutions in the area of collaborative visualization, which unfortunately uses frameworks that are hard to use, modify
or extend with new functionality. The SMARTDOC solution is not an effort to invent a new collaborative protocol, but instead a demonstration of how an industry-standard for network gaming can be used for general collaborative data visualization. The collaborative prototype discussed in the report is based on peer-to-peer technology, which provides good service when the number of users is relatively low. However, there are scenarios where many users will participate and the amount of data traffic will increase. One possible solution to the scalability problems of peer-to-peer technology is using a client/server solution. This can be done relatively easy using other classes of the DirectPlay API. The collaborative rules must of course be rewritten to be appropriate for more than two users.

Creating collaborative environments from existing techniques such as Microsoft DirectPlay is interesting and an innovative approach for developers of collaborative visualization. A paper [9], describing the solution in this report, has been accepted as an invited paper to the Computer Graphics International (CGI) 2002 conference in Bradford UK. This confirms the academic level of the work and the interest for new innovative collaborative methods.

In this report it is described how a collaborative VolumeViewer component could be used in medical work procedures, but during the project, interest has also been expressed from the meteorological and engineering area. Collaborative methods not requiring any special equipment is of course desirable, which again confirms the need for the environments developed in the SMARTDOC research project.

SMARTDOC is a two year project and is yet only in the beginning. A lot of work still remains to accomplish the project goals and requirements. The consortium has received much positive feedback from both clinicians and representatives from different companies during the first couple of months. Hopefully, SMARTDOC components will provide interactivity and new collaborative methods that meet the demands that origin from the advances in technology.
# 9. Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Doctor</td>
<td>A visualization application for medical imaging.</td>
</tr>
<tr>
<td>ActiveX</td>
<td>ActiveX components are COM components that can be embedded in documents.</td>
</tr>
<tr>
<td>ActiViz COM</td>
<td>ActiViz COM is the COM interface of the Visualization Toolkit provided by KitWare.</td>
</tr>
<tr>
<td>Amira</td>
<td>Amira is a high performance visual programming tool for advanced 3D visualization.</td>
</tr>
<tr>
<td>Annotation</td>
<td>An annotation is a text combined with a certain point or region in a dataset.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface.</td>
</tr>
<tr>
<td>AVS</td>
<td>Advanced Visual System, partner of the SMARTDOC consortium. Specialized in visualization techniques.</td>
</tr>
<tr>
<td>COM</td>
<td>COM (Component Object Model) is a software architecture that allows applications to be built from binary software components.</td>
</tr>
<tr>
<td>CT</td>
<td>CT (Computed Tomography) is a radiological examination method that uses a series of X-ray beams from different angles around the body to create a cross section image of the real tissue distribution.</td>
</tr>
<tr>
<td>DICOM</td>
<td>Digital Imaging and Communications in Medicine (DICOM) is a standard method for transferring images and associated information between different medical devices.</td>
</tr>
<tr>
<td>DirectPlay</td>
<td>A part of the DirectX API that consists of classes for network gaming.</td>
</tr>
<tr>
<td>DirectX</td>
<td>A multimedia API from Microsoft. Consists of DirectGraphics, DirectAudio, DirectInput and DirectPlay.</td>
</tr>
<tr>
<td>DLL</td>
<td>A DLL (Dynamic Link Library) is a set of small system programs that are used by large applications to perform basic functions.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface.</td>
</tr>
<tr>
<td>Habanero</td>
<td>Habanero is a state/event collaborative framework and environment provided by NCSA.</td>
</tr>
<tr>
<td>INI</td>
<td>An INI-file is an initialization file that contains text information that an application may need during startup.</td>
</tr>
<tr>
<td>IPX</td>
<td>Internetwork packet exchange protocol.</td>
</tr>
<tr>
<td>Isosurface</td>
<td>An isosurface is a surface in space where scalar values are constant.</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td><strong>MICE</strong></td>
<td>MICE (Molecular Interactive Collaborative Environment) is a research project where collaborative technology are used to examine molecular structures.</td>
</tr>
<tr>
<td><strong>MRI</strong></td>
<td>MRI (Magnetic Resonance Imaging) is an imaging technique to produce high quality cross sectional images of the inside of the body by using magnetic energy and radio waves.</td>
</tr>
<tr>
<td><strong>NCSA</strong></td>
<td>National Center for Supercomputing Applications, University of Illinois.</td>
</tr>
<tr>
<td><strong>OpenViz</strong></td>
<td>A graphics component library provided by AVS.</td>
</tr>
<tr>
<td><strong>Orthoplane</strong></td>
<td>A part of the VolumeViewer VUI that are used to move the orthoslices.</td>
</tr>
<tr>
<td><strong>Orthoslice</strong></td>
<td>A cross sectional slice of volume dataset.</td>
</tr>
<tr>
<td><strong>PACS</strong></td>
<td>Picture Archiving and Communication Systems.</td>
</tr>
<tr>
<td><strong>RAW</strong></td>
<td>RAW-files contains primitive binary data, e.g. pixel values.</td>
</tr>
<tr>
<td><strong>SVV</strong></td>
<td>SVV (Smartdoc VolumeViewer) is the file extension of the VolumeViewer parameter files.</td>
</tr>
<tr>
<td><strong>TCP/IP</strong></td>
<td>Transmission Control Protocol/Internet Protocol.</td>
</tr>
<tr>
<td><strong>VB</strong></td>
<td>Microsoft Visual Basic programming environment.</td>
</tr>
<tr>
<td><strong>VUI</strong></td>
<td>Visual User Interface.</td>
</tr>
</tbody>
</table>
10. Acknowledgements

This study has been a part of the EC funded project SMARTDOC IST 2000 28137 and was accomplished from September 2001 through February 2002. It is the final work for achieving a Master of Science Degree at the Department of Science and Technology, Linköping University.

We would like to give special thanks to Prof. Mikael Jern for support and enthusiastic coaching during the work process and for letting us participate in the project instead of merely being a usable resource.

We would also like to thank the project partners for motivating discussions and collaborative work, especially Mr. Dirk Schabe, Prof. Alan Jackson, Mr. Massimiliano Peri and the people at AETmed in Genova, Italy.

Many thanks to Mr. Lars Wigström, Miss Lisa Lindfors and Miss Hanna Lindmark for providing good ideas and suggestions during the assessment work.

Finally, we would like to thank Prof. Anders Ynnerman for arranging locals and equipment and Mrs. Lesley Bornhöft for dealing with our traveling problems.
11. References


[17] http://mice.sdsc.edu/site/, MICE project official website, 2002-02-26
12. Appendix A

Smartdoc

IST – 2000 28137

VolumeViewer Manual
Introduction

VolumeViewer is an application component for advanced 3D visualizations of MRI, CT, scientific and industrial 3D volume data. VolumeViewer creates interactive 3D surface models from 2D cross-section images on PC desktops and laptops.

A 3D surface is rendered using AVS’ state-of-the-art surface rendering method. Parameters such as cutting planes (orthoplanes), isovalue, surface opacity, viewing angle, colors, cropping and display settings, can be adjusted interactively. Volume data is in this demo version limited to AVS field data (in this prototype), a binary data format with the extension “.dat”. Two sample data sets, “ear.dat” and “hydrogen.dat”, are attached to this installation environment.

Sub windows and viewports

VolumeViewer has three viewports, which are displayed in the sub windows (A, B, and C) (figures 1a and 1b).

The 3D viewport (initially placed in sub window C) contains the volume data represented as an isosurface and three movable orthoplanes. Two 2D viewports (initially located in A and B) shows projected orthoplanes. The orthoplane with a normal in the z-direction (xy – plane) is always displayed in one of these viewports while the other viewport displays the xz- or the yz-orthoplane. To decide which orthoplane to be displayed, mark or move the desired orthoplane (xz or yz) in the 3D viewport.

By right-clicking in one of the sub windows (A or B), the corresponding 2D image is moved to C and enlarged.

3D Isosurface

Volume data is represented and displayed as an isosurface. The following parameters are used to control its appearance (figure 2):

- The “Isosurface Color” box controls the color of the isosurface. Click in the box and select a new color from the dialog box that appears.
- The “Isovalue” slider sets the isovalue of the isosurface. The value is updated continuously and thus creates the effect of an animation.
- The “Isosurface Opacity” slider sets the opacity of the isosurface.

Orthoplanes
It is possible to slide the orthoplanes back and forth in their normal directions. This can be done in either of two ways
1. Press and hold down the left mouse button in either of the corners of the plane you want to move and then move the mouse.
2. Choose “Advanced Options” and check the “handles” check box. Now each orthoplane will have handles, which makes it easier to move and select the orthoplanes.

The appearance of the orthoplanes is adjusted by changing the following orthoplane parameters (figure 3):
- By pressing the “Set Color Map” button a dialog box is opened from which the user can choose between two predefined color maps or specify a new color map. The default color map is grey scale.
- The “Orthoplane Opacity” slider controls the opacity of the orthoplanes.

Cropping
The isosurface object can be cropped using an interactive crop box. When the crop button is pressed, a box representing the extents of the isosurface is displayed. By grabbing the handles of the box, it can be resized. The isosurface is cropped according to the box’s extents.
Visibility
You can control the visibility (turn on or off) of the Isosurface and the Orthoplanes by using the “Visibility” checkboxes (figure 4). The orthoplanes can also be turned on and off individually by using the keys 1, 2 and 3 for the x, y and z-planes respectively.

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Orthoplanes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isosurface</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 5: Controllers for the visibility of the isosurface and the orthoplanes respectively

2D slices
In addition to the 3D viewport there are two 2D viewports (as mentioned above). In these viewports slices through the volume are displayed. The slice is positioned at the same place in the dataset as its corresponding orthoplane in the 3D viewport. When the orthoplane is moved the 2D slice will move correspondingly.

Annotations
VolumeViewer supports text annotations in 3D and 2D (figure 6 and 7). The user can set, modify or remove text annotations. In the lower part of the user interface panel, the controllers for annotation controls are found (figure 8):

- The “Set 3D…” button.
- The “Set 2D…” button
- The colored box shows what color the annotations have. The default color is red.
- The “On” checkbox can be used to either hide or show the annotations.
- The “Annotation list box”. The name of each annotation is displayed here. The name of a new annotation will appear in this list when it is created. By clicking on a name, the annotation dialog box will appear where the annotation can be edited (more on this below).

Figure 6: The 3D viewport with one annotation.
New 3D annotation
To set a new 3D annotation, VolumeViewer introduces a new method based on a movable 3D marker (in green), which uses the intersection between the three orthoplanes to identify the location of the annotation in the 3D viewport. When the "Set 3D…” button is clicked, a 3D marker (glyph) in a bright green color is displayed at the intersection between the three planes (figure 9). The planes can now be moved so that the glyph marker is positioned right where the user wants to place the new 3D annotation. The caption of the "Set 3D…” button has now changed to “Set …” and now works as a confirmation button. When the position is verified and the "Set …” button is clicked, the annotation dialog box appears (figure 10).

New 2D annotation
When the user press the “Set 2D…” button the cursor will change to a pen. The user can now draw an arbitrary shape with the mouse in the topmost 2D viewport. When the mouse button is released the annotation dialog box appears (figure 10).

The annotation dialog box
The dialog box displays the position of the current annotation and the text attached to it (figure 10). If a new annotation is being set, the text field is empty and the user can specify a new text and press the "Set” button. To modify an annotation, the annotation must be clicked in the list box. When the dialog box appears the text can be edited. The change will take place when the ”Modify” button is clicked. An annotation can be removed by clicking the ”Remove” button in the dialog box.

The Advanced Options Dialog box
When the advanced Options button is clicked the advanced Options Dialog box appears (figure 11). There are five options here:
- The “Handles” checkbox. The handles on the orthoplanes can be turned on or off.
- The “Axes” checkbox. Changes the visibility of the axes in 3D viewport.
- The “Background color” box. A dialog box appears when clicking in the box from where the user can select a new background color.
- The “Directional light” button. A directional light can be turned on and off. The light source is represented as a pentahedron and can be grabbed to be repositioned.
- The store / select view combo boxes. A view is a direction from which the isosurface object is viewed. It also includes the attributes of the isosurface. The user can store a view, which provides good knowledge about the dataset, and continue to interact with the volume. At a later occasion, it is possible to select and load a stored view.

![Advanced Options Dialog box]

Figure 11: The Advanced Options Dialog box

**Storing data**
The annotations and views are stored in an .svv file, located in the same directory as the application. At startup, the application looks for a file associated with the loaded volume data.

The first time a volume is used in the application, it creates a file and writes the annotation and view data to it when the user exits the program.

**Interaction with the 3D viewport**
Press down the left mouse button and move the mouse to interact with the 3D surface in the 3D-viewport. This action will rotate the 3D surface. Holding down the ALT-key at the same time moves the 3D surface in the z-direction. Holding down the CTRL-key the 3D surface moves in the xy-direction. Holding down the SHIFT key results in a zoom operation.

Alternatively the checkboxes can be used to change the kind of interaction that is performed in the 3D viewport (figure 12). The arrow is used for rotation, the magnifying glass for zooming and the crossed arrows for panning.
Collaborative sessions

VolumeViewer also comes as a network version to provide collaborative analysis. The network communication is handled by DirectPlay 8.0a, an essential part of the DirectX API that is originally meant for network game play. Thus, an installation of DirectX 8.0a is required in order to run a VolumeViewer network session. VolumeViewer uses peer-to-peer technology, which means that every participating computer establishes direct contact with the other participants.

Connecting

When clicking the “Connect” button a dialog box appears that helps the user walk through a connect wizard.

- The user has to specify his/her name during the session and which type of service to use. The application automatically lists available services, whether it is a modem connection or a TCP/IP network. The service type must be marked to be able to proceed.

- When the choice of service is made, the user has to choose between searching for available sessions and creating a new one. The user must know the IP-address of a machine (or the computer name on a LAN) where there is a session running to be able to join one. If a new session is created, a session name must be typed in. Every session has a host migration function and can have a maximum of two participants (members). Host migration makes sure that the session can continue even if the host (i.e. the user that started the session) decides to disconnect.

- When joining a session, the application checks the size of the current volume dataset. If the dimensions or the file name do not agree, a message tells the user to disconnect and load the correct volume.

- When the application has established a connection, a dialog box appears that contains controllers for the network options (figure 13).

Collaborative rules

When the session is running the two participants are treated unequally due to the rules of collaboration.

- Only one user is able to perform operations with the application. This includes all interaction and all parameter settings. This user is recognized as the active member.

- The host is the only one that can distribute or reclaim the active membership.

- The user that does not have the active membership is known as the passive member. The passive member has no possibility to affect the session. The
only actions it can perform are to use the chat, request the active membership or disconnect.

The host has the power to decide which one of the users (members) will be able to change parameters and interact with the volume. Active members have access to every controller of VolumeViewer and the interacting tools. The passive member has his/her controllers disabled and can only view the changes done by the active member or leave the session.

The host is the active member by default. When a member that is not the host has the active membership (as distributed by the host) all controllers are enabled and this user can interact and change parameters. The controllers of the host-side are disabled. The active membership can be given back to the host or be retaken by the host.

The Network Options Dialog box

At the top, the dialog box displays the current user name and the current number of participants. Below this information is a list box and the two buttons, “Approve request” and “Host Active” (the “Host active” button is replaced by the “Request Active” button if the user is not the host).

The list box contains the names of the users participating in the session. By clicking on a name, the host gives the selected user the active membership. The controllers of the selected user are enabled and the controllers of the host are disabled. When a user is given the active membership his list box becomes clickable so that the active membership can be returned to the host. The host can always retake the active membership by clicking on his/her name in the list box.

Next to the listbox there are two buttons. A passive member uses the “Request active” button when he/she wants the active membership. The host is alerted by a message in the chat box. When the host is receiving this message the “Approve Request” button can be used. It is also possible to click on the name in the participant list. If the host wants to retake the active membership, the “Host Active” button can be used.
The Dialog box also contains two chat text boxes. The one placed at the bottom is a chat window where chat messages and other general information are displayed. The user can type in chat messages and send them to the other user using the top chat text box. The messages are sent when the <ENTER> key is pressed.

**The Normalize button**
The network version of VolumeViewer has an additional button, the normalize button. It is used to translate and rotate the volume to the starting position.

**Annotations when collaborating**
All the parameters of VolumeViewer are treated almost the same as when the application is running in solitaire. The annotations are, however, treated a bit differently.

When the second user joins an already started session, all of the host-parameters are sent to the new member. This ensures that all the starting values are the same. The annotations from the host-side, the “member-annotations”, can only be modified by the host, while the user still can modify his/her own annotations. These annotations are listed in the annotation list box but can be separated from users own annotations by the preface “<Member>”.

The user that joined a session can only send new annotations to the host by setting a new annotation. Already created annotations cannot, at present date, be transferred to the host-side.

When disconnecting, all annotations in the list box are merged together in a single editable annotation component.

**Key command list**

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Turn the visibility of the X orthoplane on or off</td>
</tr>
<tr>
<td>2</td>
<td>Turn the visibility of the Y orthoplane on or off</td>
</tr>
<tr>
<td>3</td>
<td>Turn the visibility of the Z orthoplane on or off</td>
</tr>
<tr>
<td>PgUP</td>
<td>Move the last selected orthoplane one step at a time. The plane is moved towards the maximum axis value.</td>
</tr>
<tr>
<td>PgDown</td>
<td>Move the last selected orthoplane one step at a time. The plane is moved towards the minimum axis value.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>Combined with mouse moves, this key is used to zoom in the 3D viewport</td>
</tr>
<tr>
<td>ALT</td>
<td>Combined with mouse moves, this key is used to move the volume closer or farther away (z-direction) from the camera.</td>
</tr>
<tr>
<td>CTRL</td>
<td>Combined with mouse moves, this key is used to translate the volume in the X- or Y- directions.</td>
</tr>
</tbody>
</table>