CAVASS: a Computer Assisted Visualization and Analysis Software System - Visualization Aspects

George Grevera[*ab], Jayaram Udupa[b], Dewey Odhner[b], Ying Zhuge[b], Andre Souza[b], Tad Iwanaga[b], and Shipra Mishra[b]
[a]Department of Mathematics and Computer Science, Saint Joseph's University, 5600 City Avenue, Philadelphia, PA 19131;
[b]Medical Image Processing Group (MIPG), Department of Radiology, University of Pennsylvania, 423 Guardian Drive, 4th Floor Blockley Hall, Philadelphia, PA 19104-6021
*ggrevera@sju.edu; phone 1 610 660-1535; fax 1 610 660-3082; www.sju.edu/~ggrevera

ABSTRACT

The Medical Image Processing Group (MIPG) at the University of Pennsylvania has been developing and distributing medical image analysis and visualization software systems for a long period of time. Our most recent system, 3DVIEWNIX, was first released in 1993. Since that time, a number of significant advancements have taken place with regard to computer platforms and operating systems, networking capability, the rise of parallel processing standards, and the development of open source toolkits. The development of CAVASS by our group is the next generation of 3DVIEWNIX. CAVASS will be freely available, open source, and is integrated with toolkits such as ITK and VTK. CAVASS runs on Windows, Unix, and Linux but shares a single code base. Rather than requiring expensive multiprocessor systems, it seamlessly provides for parallel processing via inexpensive COWs (Cluster of Workstations) for more time consuming algorithms. Most importantly, CAVASS is directed at the visualization, processing, and analysis of medical imagery, so support for 3D and higher dimensional medical image data and the efficient implementation of algorithms is given paramount importance. This paper focuses on aspects of visualization. All of the most of the popular modes of visualization including various 2D slice modes, reslicing, MIP, surface rendering, volume rendering, and animation are incorporated into CAVASS.

Keywords: visualization, surface rendering, volume rendering, 3D imaging, software systems

1. INTRODUCTION

Our group has a long history (dating back to the 1970s) of developing and distributing with source code software systems specifically for the Computer Aided Visualization and Analysis (CAVA) of 3D and higher dimensional medical imagery. In 1980, we brought out the first ever such package for medical 3D CAVA [1]. This software executed on a Data General minicomputer, which drove a Comtal image display frame buffer. In 1982, we brought out a significantly expanded version of this software package [2]. In spite of its high machine and display device dependency, this package was distributed to over 150 sites with source code worldwide. This package was also incorporated into the General Electric CT/T 8800 scanner [3]. We subsequently developed a more advanced package [4] for the GE 9800 CT scanner. GE distributed widely these on-the-scanner packages. Earlier, we implemented DISPLAY and DISPLAY82 at the Mayo Clinic whose investigators used these packages until they started developing the Analyze system [5] around 1984-85.

Around 1987, we started the development of a Unix-workstation-based software system named 3DVIEWNX [6] which was based on standard C language and a graphical user interface library developed by us based on X Windows. It also incorporated a multidimensional generalization [7] of the 2D DICOM image representation standards. This issue of the need to handle a multidimensional vectorial image as a single entity and also to handle non-image structure information such as surfaces is only now being looked into by the standards committees related to DICOM. These issues were addressed in 3DVIEWNIX in the early stage of its design during 1987-1990. 3DVIEWNIX has incorporated numerous advanced 3D (and higher dimensional) CAVA operations including various methods of interpolation, filtering, segmentation, registration, algebraic and morphological operations, visualization methods for surfaces and volumes,

interactive structure editing and manipulation, and scene intensity and structure-based quantitative analysis. Its binary version is available freely via Internet and has been used by 100s of sites, and the source-code-version has been distributed to over 180 sites worldwide to date. We continue to maintain, distribute, and develop 3DVIEWNIX by incorporating into it all functions that we find useful after rigorously testing them in one or more of our on-going applications. About 60-person years of work has gone into 3DVIEWNIX so far. Its design has stood the test of time and of over 15 applications pursued by us since its release. Recently, we have developed specialized packages based on 3DVIEWNIX that are suitable for use in a clinical environment. Five such examples are 3DVIEWNIX-MS, for MS image analysis, 3DVIEWNIX-TV for brain tumor (volumetry) MR image analysis, 3DVIEWNIX-AVS for MRA visualization and analysis and artery/vein separation, 3DVIEWNIX-CTC for CT colonography, and 3DVIEWNIX-AIRWAY for the study of upper airway disorders in children. We also make available extensive documentation such as the 3DVIENIX Data Specification, User's manual, and Programmer's Reference manual.

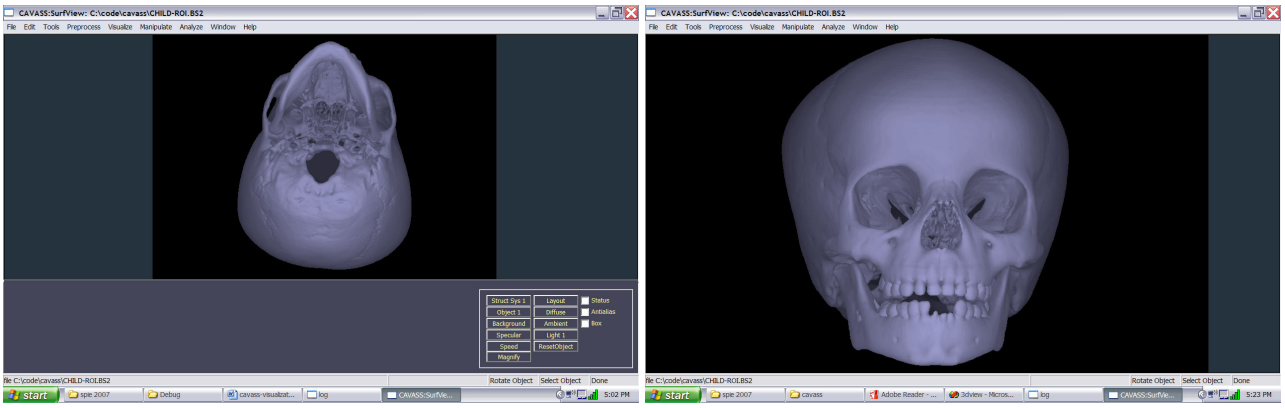

Fig. 1. Examples of triangulated shell (t-shell) rendering in CAVASS on the Windows operating system.

We have utilized the CAVA operations implemented in 3DVIEWNIX in many applications over the years. In the following ten currently ongoing applications, 100s, and in some applications 1000s, of datasets have been routinely processed. Kinematic analysis of joints [8-15] and the study of joint cartilage for the investigation of osteoarthritis [16], both via MRI: In the former, our goal is to study in vivo the normal kinematics of joints (tarsal, ankle, and glenohumeral), how these are affected by joint pathologies and soft tissue injuries, and how surgical procedures are effective in restoring normal function. In the latter, our aim is to understand the MR imaging and morphological properties of the joint cartilage in an attempt to study osteoarthritis and its treatment effects. MS [17-28] and late-life depression [29,30]: Our aim here is to study via MRI the natural course of the disease, to obtain quantitative, sensitive, and specific MRI-based markers to characterize the severity of the disease with the eventual goal of using these markers to replace the subjective test scores that are currently used, and to monitor treatment effects. Brain tumor [31,32]: The aim here is to obtain quantitative measures of edema and the active parts of gliomas via MRI to help in managing patients and to monitor treatment effects. (The software package 3DVIEWNIX-TV specially devised for this application is currently being evaluated in an American College of Radiology Imaging Network (ACRIN) trial.) Breast density [33]: The goal of this application is to obtain reproducible and accurate measures of breast fibroglandular density via digitized or digital mammograms. This measure is considered to be useful as an indicator of breast cancer risk. MRA/CTA [34-37]: The goal here is to visualize vessels via MRA/CTA free of other structures such as bones in CTA and obscuring vessels in MRA (veins while visualizing arteries, and vice versa). CT Colonography [38,39]: This modality of visualizing the colon for detecting polyps is an active area of research currently. Lung perfusion/ventilation study [40]: The goal of this application is to use helium MRI and Gadolinium-enhanced MRI of the lungs to obtain separate perfusion (Gd MRI) and ventilation (helium MRI) images and to segment and register the lungs and pulmonary vessels to obtain regional ventilation/perfusion ratios in the lungs to study various types of lung diseases. Upper airway analysis in children [41,42]: The aim here is to analyze via MRI the architecture of the upper airway and surrounding organs including soft palate, adenoid, and tonsils in children with obstructive sleep apnea.

Since the time 3DVIEWNIX was first released (1993), a number of significant developments have occurred. Most significantly, PC platforms (and the Windows OS) have gained in capability accompanied by precipitous price reductions. They have supplanted traditional Unix-based workstations as the scientific workstations of choice. Second, network connectivity (speed) has greatly increased. Third, viable parallel processing standards have been developed and are now freely available for all popular platforms and operating systems. Fourth, platform independent windowing APIs, some of which maintain the native look and feel, have been defined and implemented. And finally, toolkits such as ITK and VTK have been developed and are freely available. Although not complete applications in themselves, these toolkits provide a breadth of techniques and can be employed as building blocks of applications.

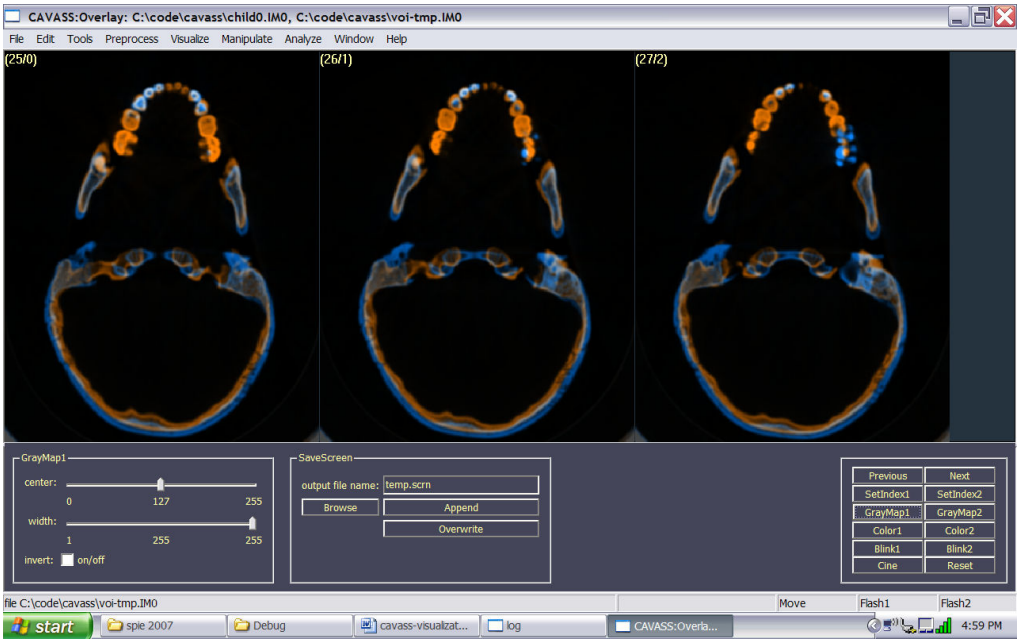

Fig. 2. An example of overlaid slice display in CAVASS on the Windows operating system.

The development of CAVASS (Computer Assisted Visualization and Analysis Software System) by our group is the next generation of 3DVIEWNIX. CAVASS is a freely available, open source software system that runs on all popular platforms, provides for efficient and parallel implementations of important image processing, visualization, and analysis algorithms, and integrates with popular toolkits. In this paper, we focus on the visualization aspects of CAVASS. All of the most of the popular modes of visualization including various 2D slice modes, reslicing, MIP, surface rendering, volume rendering, and animation are incorporated into CAVASS. Surface rendering in CAVASS utilizing digital shell and triangulated shell (t-shell) rendering algorithms are implemented in CAVASS (see Fig. 1). These in software have been shown to operate 6 to 30 times faster than hardware-based rendering. Because of their ultra high speed, they have been implemented only in sequential and not parallel mode. Volume rendering based on shell rendering has been implemented in parallel mode and has been compared to the implementation in VTK. CAVASS operates at least as fast as VTK and often achieves superior performance by a factor of 1.5 to 5.

2. METHODS

CAVASS is an open source system written entirely in C/C++ and is based on our years of experience with 3DVIEWNIX. It encompasses four groups of operations: image processing (including region of interest, interpolation, filtering, segmentation, registration, morphological operations, and algebraic operations), analysis (various methods for extracting quantitative information), visualization (including slice, reslice, maximum intensity projection, surface

rendering, and volume rendering), and manipulation (for surgical planning and simulation). In this paper, we focus on visualization operations. CAVASS retains much of the architecture of 3DVIEWNIX which has proven to be very effective, efficient, and easy to maintain and expand. The program libraries are compartmentalized into four groups: (1) data interface, (2) graphical interface, (3) process interface, and (4) CAVA (Computer Assisted Visualization and Analysis) functions. CAVA functions are further divided into four groups according to the four elements of CAVA: (a) image processing, (b) visualization, (c) manipulation, and (d) analysis. One may develop their own applications based on these libraries. In addition to these libraries, CAVASS also provides a sophisticated menu-drive GUI which together form a complete suite of medical imaging applications. Most of the popular modes of visualization including various 2D slice modes (see Fig. 2), reslicing, MIP, surface rendering (see Fig. 1), volume rendering, and animation are incorporated into CAVASS.

Our work on the surface rendering method of visualization dates back to the early days of CT and MR imaging [1,43-45]. We have devised digital surface rendering algorithms [46] that run on PCs 16-31 times faster than methods based on rendering triangulated surfaces by using hardware rendering engines [47] and take about an order of magnitude less storage space. The simplicity and efficiency of these algorithms afforded by the simplicity of the geometry of digital surfaces can also be extended to triangulated surfaces and thereby achieve an 8-10 fold speedup in software on PCs over hardware rendering engines if the triangles are embedded in a digital grid as in the output produced by the Marching Cubes family [48] of algorithms. This also affords compact storage of such surfaces. Due to this computational/storage efficiency, the need for triangle decimation methods currently pursued to reduce the number of triangles in the surface for overcoming computational bottlenecks is obviated.

For volume rendering, we developed a paradigm called shell rendering [49]. The basic idea of this approach is to represent tissue interfaces as shells and do volume rendering by projecting voxels in the shell in a back-to-front or front-to-back order onto the projection plane, and performing in the process the basic operations of volume rendering such as reflection, emission, and transmission. In one extreme, the shell may be very thin, just one voxel thick, in which case shell rendering reduces to the digital surface rendering method referred to above. In another extreme, the shell may include the whole foreground of a 3D image. In practice, the thickness of the shell is in between the two extremes. Recently, a method of volume rendering that has become popular is shear-warp rendering [50]. Like shell rendering, the shear-warp method can be used in both surface and volume mode. The speed of its surface mode is about the same as that of shell rendering in surface mode, but its volume mode is faster (about 2 times) than shell rendering [51], although the shear-warp method requires about 6-8 times more storage space than shell rendering. We have developed a new method, called shear-warp shell rendering, which combines the advantages of both methods [51] to achieve the speed of shear-warp and storage efficiency close to that of shell rendering.

Partial volume effects pose challenges to image processing operations. In volume/surface rendering, they cause severe obscuration of the objects of interest because of voxels not belonging to the object of interest exhibiting the same intensity characteristics as those belonging to the object. For example, in CT images, voxels in the bone-to-air and bone-to-fat interface appear exactly like voxels in soft tissues. We have devised a method based on scale [52], which overcomes this problem remarkably well so that the resulting renditions portray fine details significantly better than when renditions are created without using this method.

One of the earliest papers to suggest the use of structure information derived from images for surgery planning was [53]. 3DVIEWNIX contains extensive tools for manipulating (cutting, separating, mirror reflecting, moving, repositioning) structures interactively, all implemented without depending on specialized hardware, and to carry out these manipulative operations on structures defined in a hard as well as a fuzzy manner [46,54].

Surface rendering in CAVASS utilizing digital shell and triangulated shell (t-shell) rendering algorithms are implemented in CAVASS. Since these provide close to real-time speed even for extremely large surfaces, they are implemented in sequential mode only. Volume rendering (based on shell rendering), however, is implemented in parallel mode for COWs (in addition to sequential mode). The systematic row, column, and slice order of accessing image data makes it possible to parallelize shell rendering by chunks where each chunk is a set of contiguous slices. The image is divided into chunks, and each chunk is assigned to a processor which creates a mini rendered image and returns it to the manager processor together with other buffers such as z-buffer, opacity buffer, and gradient buffer. The manager processor combines all mini renditions into a final image by using the information in the associated buffers.

2.1 Parallel visualization

Parallel operations in CAVASS are implemented using the MPI (Message Passing Interface) standard [55,56]. MPI is a standard component of the Linux software distribution and is also freely available for Windows. MPI is implemented as function calls in a programming library and treats a network of computers as a distributed memory, multicomputer parallel processing system by passing messages between the systems.

Fig. 3. Head mounted display employed by CAVASS for stereo viewing.

From the perspective of the ease of parallelizability, CAVA operations may be divided into three groups, which we will call Type 1, Type 2, and Type 3. Our general approach to parallelized implementation of key CAVA operations is to perform what we call chunking. A chunk is the data contained in a contiguous set of slices. A chunk may represent SCENE or STRUCTURE data. In the former case, it represents a set of contiguous slices of the given image. In the latter case, it represents structure data contained in a contiguous set of slices. (The SCENE data type represents nD images - scalar, vector-valued or binary with a regular (rectangular grid) or arbitrary sampling scheme. STRUCTURE data type represents multidimensional non-image structure information usually derived from SCENE data.) There are many operations in CAVA, which work, or, which can be made to work, in a more-or-less "slice-by-slice", and hence in a "chunk-by-chunk", manner. In these operations, a slice (or chunk) worth of data needs to be accessed only once to complete the operation (or to complete one iteration of the operation) and produce the final output. Such operations are labeled Type 1. Examples of such operations are: image gray level slice interpolation methods (linear, spline-based methods) [57], shape-based (binary as well as gray-level) interpolation [57-61], image-based registration (via mutual information/correlation) [62,63], diffusive filtering [64-66], inhomogeneity correction [67], all non-user-steered slice-by-slice segmentation methods (such as clustering techniques), non-connected isosurface detection, and structure manipulation [46,54]. There are other CAVA operations, which work (chunk-by-chunk) in the above sense but some further operation is needed to combine the outputs produced by the chunks to yield the final output. Such operations are labeled Type 2. These are more difficult to parallelize and implement than Type 1 operations. Examples of such operations are various surface and volume rendering methods, particularly those that use some sort of a front-to-back or back-to-front splatting/projection strategy, such as shell and shear-warp rendering methods [46,49,50,51]. We label those CAVA operations, which require each slice/chunk to be accessed more than once to complete the operation as Type 3. These are more difficult than Type 1 and Type 2 operations to parallelize the implementation. These operations can be characterized by graph traversal methods and the number of times a slice (chunk) is accessed depends on the shape of the objects represented in the image and on the orientation of the slices with respect to the object. Examples of such operations are: connected isosurface detection [68-70], connected object segmentation in a hard or fuzzy manner [71-81], and optimal path (graph cut) and fast marching (level set) methods of segmentation [82,83]. In connected isosurface detection [68,69,84], for example, the average number of accesses of an axial slice in a 3D image of the human body is typically in the range 1.5-1.8. Our aim in CAVASS is to parallelize the implementation for the following 10 groups of key CAVA operations: gray-level slice interpolation, shape-based interpolation, image-based registration

(via mutual information, correlation), diffusive filtering (scale-based and non-scale-based), inhomogeneity correction (scale-based), structure manipulation (hard and fuzzy [46,54]), surface and volume rendering (via shell and shear-warp techniques), connected isosurface detection (both digital and triangulated), and fuzzy connectedness segmentation. With regard to visualization, the Type-2 operations we have considered are mainly those related to surface and volume rendering. We will describe our parallelization strategy for shell rendering in the surface mode. Minor modifications of this strategy will work for other methods in this group. This description is applicable to 3D surfaces. For higher dimensional surfaces, their 3D cross sections have to be produced first.

Table 1. Description of datasets of varying sizes used in the comparisons.

dataset name	voxel size	image size	data size
regular	0.98 x 0.98 x 3.00 mm	256 x 256 x 46	6 MB
large	0.68 x 0.68 x 1.50 mm	512 x 512 x 459	241 MB
super	0.24 x 0.24 x 0.50 mm	1023 x 1023 x 417	873 MB

The main theorem [85] underlying the operation of shell rendering (and of any voxel splatting (projection) method) states a rule for the order in which the voxels should be accessed and projected so that the voxels projecting onto any given pixel in the projection plane are in a strict back-to-front order. The theorem essentially says that this order of access is one of eight possible orders which result when all possible permutations of indexing the rows, columns, and slices of a 3D rectangular array are considered wherein the indices are allowed to change from the beginning to end and vice versa. Each of the eight (2^3) orders corresponds to each of the eight octants of the 3D space in which the projection plane/viewpoint is situated. Given the position of the projection plane/viewpoint, the indexing order can be determined by table lookup [46,49,86]. To render a 3D surface (digital or triangulated) via the shell rendering paradigm, a special data structure called a shell is employed which stores the surface elements (and associated descriptions such as normal, neighborhood configuration) in a row-by-row and slice-by-slice manner. Any surface element (a voxel or oriented voxel face in the case of a digital surface, and a triangle in the case of a triangulated surface) in any given row and slice can be accessed from the shell almost at the speed of random access of a voxel from a 3D rectangular array. Our overall parallelization strategy consists of the following four steps: set up, dividing into chunks, rendering each chunk, combining the mini-renditions.

Step 1 - Set up: This consists of view-independent and view-dependent set up operations. Most operations (geometric transformation, voxel/triangle projection, shading calculation) in shell rendering are reduced to table look up operations because of the regular geometry of the digital data (this is also true for triangles embedded in cubes as in the Marching Cubes methods). Such tables are view-dependent. View-independent tables are set up only once.

Step 2 - Dividing into chunks: Each chunk corresponds to surface (shell) data within a set of contiguous slices. Breaking up of the shell into chunks can be done very inexpensively, since the shell structure is slice-based.

Step 3 - Rendering each chunk: This is where most of the computations come from. Each processor renders its own chunk to create a mini-rendition.

Step 4 - Combining the mini-renditions: Knowing the location of the projection plane/viewpoint, the same table look up mechanism described earlier can be utilized to determine the order in which the mini-renditions are to be combined. "Combining" in the case of surface rendering simply means determining which pixel's rendered value in the mini-renditions should survive in the final rendition when pixels overlap. In other words, this involves determining which among the overlapping pixels should overwrite (as per the back-to-front strategy) the rendered values. In volume rendering, this involves determining the proper order of composting the mini-renditions to handle the emission, reflection, and transmission components of rendering calculations.

A front-to-back strategy can also be developed along similar lines. In our experience with single processor implementation in 3DVIEWNIX [46,49], this strategy yields slightly faster renditions than back-to-front. However, we have used the latter combined with z-buffer in 3DVIEWNIX because of their use in other operations.

We have chosen to implement our parallel algorithms using the MPI/OpenMPI standard which is commonly and freely available for Linux, Unix, and Windows. Please note that MPI or OpenMPI should not be confused with MP or OpenMP [87,88]. OpenMP (Open specifications for Multi Processing) is a parallel processing standard for "multi-threaded, shared memory parallelism" [87]. OpenMP requires special compilers that recognize compiler directives embedded in the source code to control parallelism. Furthermore, "OpenMP is not meant for distributed memory parallel systems" [87]. Typically, OpenMP systems are expensive, tightly coupled shared memory multiprocessor systems such as the SGI Origin systems or the new SGI Altix 4700 which "supports up to 512 processors under one instance of Linux and as much as 128TB of globally shared memory" [89]. Our approach uses inexpensive, commonly available "commodity" workstations/PCs.

Another area where parallelism can be employed is in stereo rendering for displays such as those shown in Fig. 3 [90]. The CAVASS stereo surface/volume rendering implementation renders from not one but two different points of view (one for each eye) for each given position of the projection plane. Typically the angle between the two nearby viewpoints is about 4°. In CAVASS, this number is a parameter whose value can be modified according to an individual's vision characteristics. The graphics interface library and the GUI handles stereo display hardware devices such as the one in Fig. 3. Library functions support all necessary interactions with the stereo display, including pointing to locations on the structures in their surface/volume renditions (we have previously published such algorithms (68, 69), interactively performing curved cuts, repositioning of segments, and making linear, angular, and curvilinear measurements interactively.

2.2 Portable user interface

To implement a portable GUI, we considered Qt [91], wxWidgets (formerly called wxWindows) [92-95], and FLTK (Fast Light Tool Kit) [96]. Qt was eliminated from further consideration as it is proprietary/closed and requires fees ($2330 for one developer on a single platform (i.e., Unix, Windows, or Mac OS); $4660 for one developer on 3 platforms). FLTK and wxWidgets both provide a common C++ API that one may use to develop portable GUIs. Both are freely available on a variety of platforms, are open source, support OpenGL, drag and drop, and cut and paste in a platform-independent manner. Choosing between FLTK and wxWidgets is not simple but we feel that wxWidgets is superior because it endeavors to maintain the native look and feel of the platform on which it is running. Furthermore, only wxWidgets supports printing in a platform-independent manner. For these reasons, we chose wxWidgets for implementing the GUI in CAVASS.

Table 2. Surface rendering timing comparison for CAVASS (sequential implementation with and without antialiasing) and surface rendering as implemented in VTK.

dataset name	CAVASS seq/no aa	CAVASS seq/aa	VTK
regular	0.03	0.06	0.29
large	0.11	0.19	0.41
super	0.16	0.26	1.38

3. RESULTS

The two major volume visualization methods of surface rendering and volume rendering have been implemented and tested in CAVASS. We compared the implementations of sequential t-shell surface rendering implemented entirely in software in CAVASS with hardware-assisted surface rendering using the Marching Cubes method as implemented in VTK (using the vtkImageMarchingCubes class). We also compared sequential and parallel volume rendering implemented entirely in software in CAVASS with two methods of volume rendering (ray casting and 2D texture mapping) implemented in VTK (using the vtkVolumeRayCastMapper and vtkOpenGLVolumeTextureMapper2D classes, respectively). The timing results in seconds per frame were obtained by applying the various visualization techniques to three datasets of varying sizes (regular or clinically typical, large, and super) as shown in Table 1 and by creating 10s of frames and averaging the required time. Results for sequential surface rendering and parallel and

sequential volume rendering appear in Tables 2 and 3, respectively. Table 2 demonstrates that sequential CAVASS shell rendering, entirely in software and without antialiasing, was more than 8.5 times faster than hardware-based rendering as implemented in VTK for the largest dataset (super) in our test. With antialiasing, CAVASS shell rendering was more than 5 times faster. For volume rendering, Table 3 shows that the CAVASS implementation, entirely in software, was faster than both ray casting and 2D texture mapping as implemented in VTK for both the regular and large datasets. For the super dataset, sequential CAVASS volume rendering was slower than volume rendering in VTK but the parallel implementation of volume rendering in CAVASS was almost twice as fast as ray casting in VTK. Although VTK ray casting was able to render the largest dataset, 2D texture mapping as implemented in VTK was unable to render the largest dataset after more than 240 seconds. This is likely due to the limited amount of memory on the graphics card. When we compare VTK ray casting to VTK 2D texture mapping, we note the trend that VTK ray casting is consistently faster than VTK 2D texture mapping. Since CAVASS parallel volume rendering is consistently faster than both VTK ray casting and VTK 2D texture, we conclude that even with additional video memory, CAVASS parallel volume rendering would be faster than VTK 2D texture rendering of the largest dataset.

All sequential tests were performed on a Dell dual processor, 3.4 GHz Xeon system with 4 GB RAM and hyperthreading enabled under the Linux operating system version 2.6.9-1.667smp. (It is interesting to note that with hyperthreading enabled, Linux reports the presence of 4 CPUs in /proc/cpuinfo.) The parallel tests were performed on a cluster of six single processor systems (Dell single processor, 3.6 GHz Pentium systems with 3 GB RAM and hyperthreading enabled under the Linux operating system version 2.6.9-1.667smp) interconnected by an inexpensive 1Gb (gigabit) switch (Dell PowerConnect 2608, an 8-port 1-gigabit Ethernet switch). All systems had Nvidia Quadro NVS280 PCIe 64 MB video cards.

Table 3. Volume rendering timing comparison (in seconds) for sequential and parallel implementations of CAVASS shell rendering, VTK ray casting, and VTK 2D texture mapped volume rendering.

	CAVASS		VTK	
dataset name	**sequential**	**parallel**	**ray casting**	**2D texture**
regular	0.56	0.06	1.09	1.20
large	3.53	1.36	5.03	18.32
super	9.77	3.66	6.94	>240.00

4. CONCLUSIONS

We described the visualization methods implemented in CAVASS, a new open source, open platform software system which is the next incarnation of the previously established and widely distributed 3DVIEWNIX software system. We demonstrated the extremely efficient implementation of visualization algorithms in sequential and parallel modes on COWs in CAVASS. CAVASS is the only freely available, open source image processing, analysis, and visualization software system for multidimensional medical imagery that incorporates other open source toolkits and provides for the efficient and parallel implementations of important CAVA algorithms. In particular and with regard to visualization, surface rendering in CAVASS entirely in software was demonstrated to be more than 8.5 times faster than hardware-assisted surface rendering in an established open source system, VTK. For volume rendering, we demonstrated that sequential volume rendering in CAVASS entirely in software is faster for the regular and large datasets in our test, and for the largest dataset (super), parallel volume rendering in CAVASS was almost twice as fast as the fastest hardware-based method in VTK.

CAVASS is available from www.mipg.upenn.edu/~cavass.

ACKNOWLEDGEMENT

The authors gratefully acknowledge support for this work from DHHS grant EB004395.

REFERENCES

1. J.K. Udupa: "DISPLAY - A system of programs for two- and three-dimensional display of medical objects from CT data," Technical Report MIPG41, Medical Image Processing Group, Department of Computer Science, SUNY/Buffalo, Buffalo, New York, 1980.
2. J.K. Udupa: "DISPLAY82 - A system of programs for the display of 3D information in CT data," Technical Report MIPG67, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, April 1983.
3. L.S. Chen, G.T. Herman, C.R. Meyer, R.A. Reynolds, and J.K. Udupa: "3D83 - An easy-to-use software package for three-dimensional display from computed tomograms," Proceedings of IEEE Computer Society International Symposium on Medical Images and Icons, Arlington, Virginia, pp. 309-316, 1984.
4. J.K. Udupa, G.T. Herman, P.S. Margasahayam, L.S. Chen, and C.R. Meyer: "3D9S: A turnkey system for the display and analysis of 3D medical objects," SPIE Proceedings 671:154-168, 1986.
5. R.A. Robb, D.P. Hanson, R.A. Karwoski, A.G. Larson, E.L.Workman, and M.C. Stacy: "ANALYZE: A comprehensive, operator-interactive software package for multidimensional medical image display and analysis," Computerized Med. Imag. Graph. 13:433-454, 1989.
6. J.K. Udupa, D. Odhner, S. Samarasekera, R. Goncalves, K. lyer, K. Venugopal, and S. Furuie: "3DVIEWNIX: an open, transportable, multidimensional, multimodality, multiparametric imaging software system," in SPIE Proceedings 2164:58-73, 1994.
7. J.K. Udupa, H.M. Hung, D. Odhner, and R. Goncalves: "Multidimensional data format specification: A generalization of the American College of Radiology National Electric Manufacturers Association Standards," Journal of Digital Imaging 5(l):26-45, 1992.
8. E. Stindel, J. Udupa, B. Hirsch, D. Odhner, and C. Couture: "3D MR image analysis of the morphology of the rear foot: Application to classification of bones," Computerized Medical Imaging and Graphics 23:75-83, 1999.
9. E. Stindel, J. Udupa, B. Hirsch, and D. Odhner: "A characterization of the geometric architecture of the peritalar joint complex via MRI: An aid to classification of feet," IEEE Transactions on Medical Imaging 18:753-763, 1999.
10. J. Udupa, B. Hirsch, S. Samarasekera, H. Hillstrom, G. Bauer, and B. Kneeland: "Analysis of in vivo 3D internal kinematics of the joints of the foot," IEEE Transactions on Biomedical Engineering 45:1387-1396, 1998.
11. E. Stindel, J. Udupa, B. Hirsch, and D. Odhner: "An in vivo analysis of the peritalar joint complex based on MR imaging," IEEE Transactions on Biomedical Engineering 48:236-247, 2001.
12. B.E. Hirsch, J.K. Udupa, and E. Stindel: "Tarsal Joint Kinematics via 3D Imaging," in 3D Imaging in Medicine, J.K. Udupa and G.T. Herman (editors.), CRC Press, Boca Raton, Florida, pp. 329-359, 2000.
13. B.E. Hirsch, J.K. Udupa, R. Goncalves, and D. Roberts: "Kinematics of the joints of the foot via three-dimensional magnetic resonance images," Proceedings of the First Conference on Visualization in Biomedical Computing VBC'90:232-237, May 22-25, 1990.
14. R.C. Rhoad, JJ. Klimkiewicz, G.R. Williams, S.B. Kesmodel, J.K. Udupa, J.B. Kneeland, and J.P., lannotti: "A new in vivo technique for 3D shoulder kinematics analysis," Skeletal Radiology 27:92-97, 1998.
15. J. Woodburn, J. Udupa, B. Hirsch, R. Wakefield, P. Helliwell, N. Reay, P. O'Connor, A. Budgen and P. Emery: "The geometrical architecture of the subtalar and midtarsal joints in rheumatoid arthritis based on MR imaging," Arthritis and Rheumatism 46(12):3168-3177, 2002.
16. A. Gougoutas, A. Borthakur, A. Wheaton, J.B. Kneeland and R. Reddy: "The use of a semi-automatic segmentation strategy in computing cartilage volumes," in Proceedings of ISMRM 152-152, 2002.
17. J. Udupa, L. Wei, S. Samarasekera, Y. Miki, M. Buchem and R. Grossman: "Multiple sclerosis lesion quantification using fuzzy-connectedness principles," IEEE Transactions on Medical Imaging 16:598-609, 1997.
18. S. Samarasekera, J. Udupa, Y. Miki and R. Grossman: "A new computer-assisted method for the quantification of enhancing lesions in multiple sclerosis," Journal of Computer Assisted Tomography 21:145-151, 1997.
19. Y. Miki, R. Grossman, J. Udupa, S. Samarasekera, M. van Buchem, B. Cooney, S. Pollack, D. Kolson, M. Polansky and L. Mannon: "Computer-assisted quantitation of enhancing lesions in multiple sclerosis: Correlation with clinical classification," American Journal of Neuroradiology 18:705-710, 1997.
20. M. van Buchem, J. Udupa, F. Heyning, M. Boncoeur-Martel, Y. Miki, J. McGowan, D. Kolson, M. Polansky and R. Grossman: "Global volumetric estimation of disease burden in multiple sclerosis based on magnetization transfer imaging," American Journal of Neuroraetiology 18:1287-1290, 1997.

21. Y. Miki, R. Grossman, J. Udupa, L. Wei, D. Kolson and L. Mannon: "Isolated U-fiber involvement in MS: Preliminary observations," Neurology 50:1301-1306, 1998.
22. M. Filippi, M. Horsfield, J. Hajnal, P. Narayana, J. Udupa, T. Yousry and A. Zijdenbos: "Quantitative assessment of magnetic resonance imaging lesion load in multiple sclerosis," Journal of Neurology, Neurosurgery, and Psychiatry 64 (Supplement):S88-S93, 1998 (invited paper).
23. M. van Buchem, R. Grossman, C. Armstrong, M. Polansky, Y. Miki, F. Heyning, M. Boncoeur-Martel, L. Wei, J. Udupa, M. Grossman, D. Kolson and J. McGowan: "Correlation of volumetric magnetization transfer imaging with clinical data in MS" Neurology, 50:1609-1617, 1998.
24. M. Phillips, R. Grossman, Y. Miki, L. Wei, D. Kolson, M. van Buchem, M. Polansky, J. McGowan and J. Udupa: "Comparison of T2 lesion volume and magnetization transfer ratio histogram analysis and atrophy and measures of lesion burden in patients with multiple sclerosis," American Journal of Neuroradiology 19:1055-1060, 1998.
25. A. Kumar, Z. Jin, W. Bilker, J. Udupa and G. Gottlieb: "Late-onset minor and major depression: Early evidence for common neuroanatomical substrates detected by using MRI" in Proceedings of the National Academy of Science 95:7654-7658, 1998.
26. Y. Miki, R. Grossman, J. Udupa, L. Wei, M. Polansky, L. Mannon and D. Kolson: "Relapsing-remitting multiple sclerosis: Longitudinal analysis of MR images - lack of correlation between changes in T2 lesion volume and clinical findings," Radiology 213:395-399, 1999.
27. Y. Ge, R. Grossman, J. Udupa, L. Wei, L. Mannon, M. Polansky and D. Kolson: "Brain atrophy in relapsing-remitting multiple sclerosis and secondary progressive multiple sclerosis: Longitudinal quantitative analysis," Radiology 214:665-670, 2000.
28. Y. Ge, J. Udupa, L. Nyul, L. Wei and R. Grossman: "Numerical tissue characterization in MS via standardization of the MR image intensity scale," Journal of Magnetic Resonance Imaging 12:715-721, 2000.
29. A. Kumar, W. Bilker, Z. Jin, J. Udupa and G. Gottlieb: "Age of onset of depression and quantitative neuroanatomic measures: Absence of specific correlations," Psychiatry Research Neuroimaging 91:101-110, 1999.
30. A. Kumar, W. Bilker, Z. Jin and J. Udupa: "Atrophy and high intensity lesions: Complementary neurobiological mechanisms in late-life major depression," Neuropsychopharmacology 22:264-274, 2000.
31. G. Moonis, J. Liu, J. Udupa and D. Hackney: "Estimation of tumor volume using fuzzy connectedness segmentation of MRI," American Journal of Neuroradiology 23:356-363, 2002.
32. J. Liu, J. Udupa, D. Hackney and G. Moonis: "Brain tumor segmentation in MRI using fuzzy connectedness method," in Proceedings of SPIE 4322:1455-1465, 2001.
33. P. Saha, J. Udupa, E. Conant, D. Chakraborty and D. Sullivan: "Breast tissue glandularity quantification via digitized mammograms," IEEE Transactions on Medical Imaging 20:792-803, 2001.
34. P. Saha, J. Udupa and J. Abrahams: "Automatic bone-free rendering of cerebral aneurysms via 3D-CTA," in Proceedings of SPIE, 4322:1264-1272, 2001.
35. J. Abrahams, P. Saha, R. Hurst, P. LeRoux and J. Udupa: "Three-dimensional bone-free rendering of the cerebral circulation using computed tomographic angiography and fuzzy connectedness," Neurosurgery, 51:264-269,2002.
36. T. Lei, J. Udupa, P. Saha and D. Odhner: "Artery-vein separation via MRA - An image processing approach," IEEE Transactions on Medical Imaging 20:689-703, 2001.
37. B. Rice and J. Udupa: "Fuzzy connected clutter-free volume rendering for MR angiography," International Journal of Imaging Systems and Technology 11:62-70, 2000 (invited paper).
38. J. Udupa, D. Odhner and H. Eisenberg: "New automatic mode of visualizing the colon via cine CT," in Proceedings of SPIE 4319:237-243, 2001.
39. E. Balogh, E. Sorantin, L. Nyul, K. Palagyi, A. Kuba, G. Werkgartner and E. Spuller: "Virtual dissection of the colon - Technique and first experiments with artificial and cadaveric phantoms," in Proceedings of SPIE 4681, 2002.
40. B. Wang, P. Saha, R. Rizi, D. Robert, J. Baumgardner, M. Ishii, W. Gefter, M. Schnall, G. Johnson and J. Udupa: "Airway segmentation via Hyperpolarized He Gas MR! using scale-based fuzzy connectedness," in Proceedings of ISMRM 763-763, 2002.
41. J. Liu, J. Udupa, D. Odhner, J. McDonough and R. Arens: "Upper airway segmentation and measurement in MRI using fuzzy connectedness," in Proceedings of SPIE, 4683:238-247, 2002.
42. Arens, R., McDonough, J.M., Corbin, A.M., Rubin, N.K., Carroll, M.E., Pack, A.I., Liu, J.G., Udupa, J.K.: Upper airway size analysis using magnetic resonance imaging in children with obstructive sleep apnea syndrome, American Journal of Respiratory and Critical Care Medicine 167:65-70, 2003.

43. M.D. Altschuler, Y. Censor, P.B.B. Eggermont, G.T. Herman, Y.H. Kuo, R.M. Lewitt, M.R. McKay, H. Tuy, J.K. Udupa, and M. Yau, M.: "Demonstration of a software package for the reconstruction of the dynamically changing structure of the human heart from cone-beam x-ray projections," Journal of Medical Systems 4(2):289-304, 1980.
44. J.K. Udupa: "Display of 3-D information in discrete 3-D scenes produced by computerized tomography," Proceedings of the IEEE 71:420-431, 1983.
45. G.T. Herman and J.K. Udupa, "Display of 3-D information in 3-D digital images: Computational foundations and medical applications," IEEE Computer Graphics and Applications 3:39-46, 1983.
46. J.K. Udupa and D. Odhner: "Fast visualization, manipulation, and analysis of binary volumetric objects," IEEE Computer Graphics and Applications ll(6):53-62, 1991.
47. G.J. Grevera, J.K. Udupa, and D. Odhner: "An order of magnitude faster surface rendering in software on a PC than using dedicated rendering hardware," IEEE Transactions on Visualization and Computer Graphics 6(4):335-345, 2000.
48. W.E. Lorensen and H.E. Cline: "Marching cubes: A high resolution 3D surface construction algorithm," Computer Graphics 21(4): 163-169, 1987.
49. J.K. Udupa and D. Odhner: "Shell rendering," IEEE Computer Graphics and Applications 13(6):58-67, 1993.
50. P. Lacroute and M. Levoy: "Fast volume rendering using a shear-warp factorization of the viewing transformation," Proc. SIGGRAPH:451-458, 1994.
51. A.X. Falcao, L.M. Rocha and J.K. Udupa: "Comparative analysis of shell rendering and shear-warp rendering," SPIE Proceedings 4681:472-482, 2002.
52. A.D. Souza, J.K. Udupa, and P.K. Saha: "Volume rendering in the presence of partial volume effects," Proc. SPIE 4681:649-660, Medical Imaging 2002: Visualization, Image-Guided Procedures, and Display; S.K. Mun; Ed., 2002.
53. J.K. Udupa: "Interactive segmentation and boundary surface formation for 3-D digital images," Computer Graphics and Image Processing, 18:213-235, 1982.
54. D. Odhner and J.K. Udupa, "Shell manipulation: Interactive alteration of multiple-material fuzzy structures," SPIE Proceedings 2431:35-42, 1995.
55. www-unix.mcs.anl.gov/mpi/, The Message Passing Interface (MPI) standard
56. www.open-mpi.org/, Open MPI: Open Source High Performance Computing
57. G.J. Grevera and J.K. Udupa: "Shape-based interpolation of multidimensional grey-level images," IEEE Transactions on Medical Imaging 15(6):881-892, 1996.
58. S.P. Raya, and J.K. Udupa: "Shape-based interpolation of multidimensional objects," IEEE Transactions on Medical Imaging 9(l):32-42, 1990.
59. W.E. Higgins, C. Morice, and E.L. Ritman: "Shape-based interpolation of thin structures in three-dimensional images," IEEE Trans. Medical Imaging 12(3):439-450, 1993.
60. G.T. Herman, J. Zheng, and C.A. Bucholtz: "Shape-based interpolation," IEEE Computer Graphics and Applications 12(3):69-79, 1992.
61. G.M. Treece: "Volume Measurement and Surface Visualisation in Sequential Freehand 3D Ultrasound." Ph.D. Thesis, Cambridge University, November 2000.
62. W.M. Wells III, P. Viola, H. Atsumi, S. Makajima, and R. Kikinis: "Multi-modal volume registration by maximization of mutual information," Medical Image Analysis 1(1):35-51, 1996.
63. J.O. Lauchaud and A. Montanvert: "Continuous analogs of digital boundaries: A topological approach to iso-surfaces," Graphical Models 62(3): 129-164, 2000.
64. G. Gerig, O. Kubler, R. Kikinis, and F.A. Jolesz: "Nonlinear anisotropic filtering of MRI data," IEEE Trans. Medical Imaging ll(2):221-232, 1992.
65. P.K. Saha, and J.K. Udupa: "Scale-based image filtering preserving boundary sharpness and fine structure," IEEE Transactions on Medical Imaging 20(11): 1140-1155, 2001.
66. P. Perona and J. Malik: "Scale-space and edge detection using anisotropic diffusion," IEEE Pattern Analysis and Machine Intelligence 12(7):629-639, 1990.
67. Y. Zhuge, J.K. Udupa, J. Liu, P.K. Saha, and T. Iwanaga: "Scale-based method for correcting background intensity variation in acquired images," Proc. SPIE 4684:1103-1111, Medical Imaging 2002: Image Processing; M. Sonka, J.M. Fitzpatrick; Editors, 2002.
68. J.K. Udupa: "Multidimensional digital boundaries," CVGIP: Graphical Models and Image Processing 50(4):311-323, 1994.
69. J.K. Udupa S.N. Srihari and G.T. Herman: "Boundary detection in multidimensions," IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-4:41-50, 1982.

70. Y. Boykov, O. Veksler, and R. Zabih: "Fast approximate energy minimization via graph cuts," IEEE Transactions on Pattern Analysis and Machine Intelligence 23:1222-1239, 2001.
71. J. Udupa and S. Samarasekera: "Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation," Graphical Models and Image Processing 58:246-261, 1996.
72. P. Saha, J. Udupa and D. Odhner: "Scale-based fuzzy connected image segmentation: Theory, algorithms and validation," Computer Vision and Image Understanding 77:145-174, 2000.
73. P. Saha and J. Udupa: "Relative fuzzy connectedness among multiple objects: Theory, algorithms, and applications in image segmentation," Computer Vision and Image Understanding 82:42-56, 2001.
74. P. Saha and J. Udupa: "Fuzzy connected object delineation: Axiomatic path strength definition and the case of multiple seeds," Computer Vision and Image Understanding 83:275-295, 2001.
75. J. Udupa, P. Saha, and R. Lotufo: "Relative fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 1485-1500, 2002.
76. T. Jones: Image-Based Ventricular Blood Flow Analysis, Doctoral Dissertation, University of Pennsylvania, 1998.
77. J. Cutrona and N. Bonnet: "Two methods for semi-automatic image segmentation based on fuzzy connectedness and watersheds," France-Iberic Microscopy Congress, Barcelona, 23-24, 2001.
78. R. He and P. Narayana: "Detection and delineation of multiple sclerosis lesions in gadolinium-enhanced 3D Tl-weighted MRI data," in Proceedings of IEEE Symposium on Computer Based Medical Systems, 2000.
79. T. Aldeliesten, W. Niessen, K. Vincken, J. Maintz, F. Jansen, O. van Nieuwenhuizen, and M. Viergever: "Objective and reproducible segmentation and quantification of tuberous sclerosis lesions in FLAIR brain MR images," in Proceedings of SPIE 4322:1509-1518, 2001.
80. Y. Jin, A. Laine, and C. Imielinska: "An adaptive speed term based on homogeneity for level-set segmentation," in Proceedings of SPIE.
81. .S. Henn, M.G. Lemole, M.A.T. Ferreira, F.L. Gonzalez, M. Schornak, M.C. Preul, and R.F. Spetzler: Interactive stereoscopic virtual reality: a new tool for neurosurgical education," J. Neurosurgery 96(1): 144-149, 2002.
82. J. Sethian: Level Set Methods, Cambridge University Press, 1996.
83. S.P. Raya, J.K. Udupa, and W.A. Barrett: "A PC-based 3D imaging system: algorithms, software, and hardware considerations," Computerized Medical, Imaging and Graphics 14(5):353-370, 1990.
84. G. Frieder, D. Gordon, and R.A. Reynolds: "Back-to-front display of voxel-based objects," IEEE Computer Graphics and Applications 5:52-60, 1985.
85. G.J. Grevera, J.K. Udupa, and D. Odhner: 'T-shell rendering," SPIE Proceedings 4319:413-425, 2001.
86. A.X. Falcao, J. Stolfi, and R. Lotufo: "The image foresting transform: Theory, algorithms, and applications," IEEE Pattern Analysis and Machine Intelligence 26(1): 19-29, 2004.
87. www.llnl.gov/computing/tutorials/openMP/#Introduction, OpenMP
88. www.openmp.org/drupal/
89. www.sgi.com/products/servers/altix/4000/
90. i-glassesonline.stores.yahoo.net/iglassespc-3d.html
91. www.prolltech.com.
92. S. Cochran: "wxWindows 2.2 offers cross-platform alternative to Java," Dr. Dobb's Journal, Aug. 2000.
93. V. Zeitlin: "The wxWindows cross-platform framework," Dr. Dobb's Journal, May 2001.
94. T. Rampersad: "wxWindows for cross-platform coding," ACM Linux Journal 2003(111):6, 2003.
95. wxWidgets.org
96. www.fltk.org.