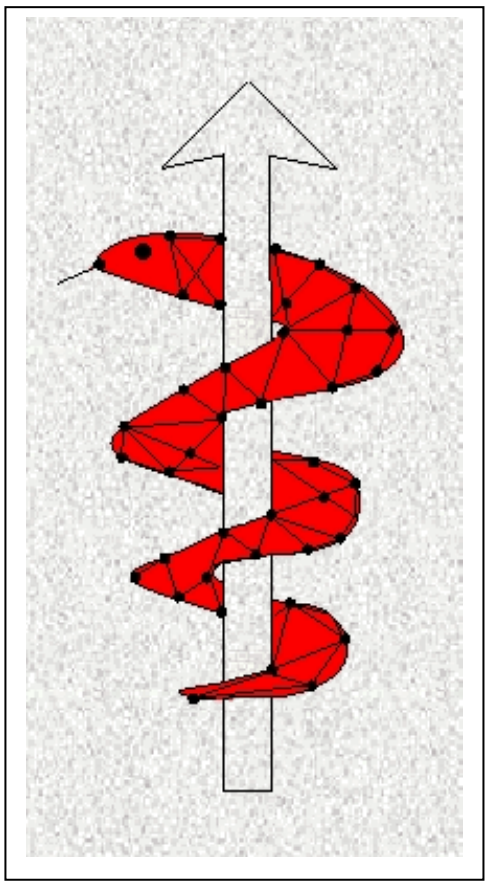
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D8Xd

Status:	Final
Version:	1.0
Security:	Public
Responsible:	caesar
Authoring Partners:	caesar

Release History

Version	Date
1.0	24.07.03

The SimBio Consortium :
NEC Europe Ltd. – UK
A.N.T. Software – The Netherlands
CNRS-DR18 – France
Sheffield University – UK
MPI of Cognitive Neuroscience – Germany
Biomagnetisches Zentrum Jena – Germany
ESI Group – France
Smith & Nephew - UK
U. Maribor - Slovenia
Teaching Hospital Maribor - Slovenia

The Julius Software Framework for Medical Visualization

1. Introduction

In image guided surgery the vision of reality is enhanced using information from CT, MR and other medical imaging data. The use of operational instruments can be guided by aligning the image data to the patients position on the operating table. In order to bring the actual computer-aided pre-operative planning scheme into the operating room, it is best to use a single software environment for both – the pre-operative planning process and the intra-operative navigation. Since image guided surgery is a relatively new field, advanced algorithms and new techniques should easily be integrated into this environment. Consequently our software framework *Julius*[1,2,3] was developed as a general platform for a complete medical pipeline, including data import, image processing, visualization and navigation (fig. 1). Owing to its platform independence, ease of extensibility and scalability it can also be used for rapid tool development by third party researchers.

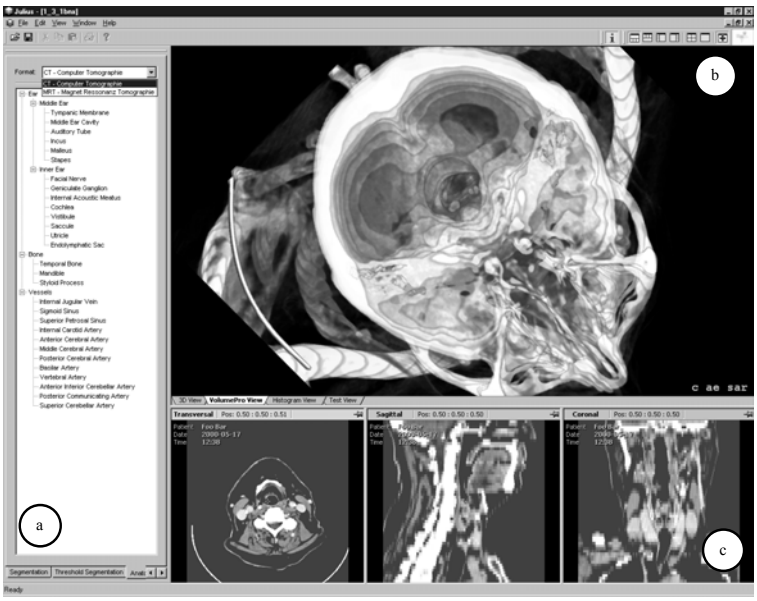


Figure 1. Screenshot of *Julius*.
(a) the *Global Bar*, (b) volume rendering, (c) reconstructed cross-sections.

1.1 Platform Independence

Platform independence is achieved by using freely available cross-platform libraries like the *Visualization Toolkit (Vtk)*[4] for image processing and rendering as well as the *Qt*[5] *Toolkit* for user interface development and file handling. Since all core- and plugin-classes are implemented in C++ the code can be used to generate executable programs on many systems. Tested operating systems are Windows NT , Windows2000, WindowsXP as well as Silicon Graphics machines under IRIX 6.5., Sun Workstations with the Solaris operating system and computers running Linux distributions (e.g. Red Hat 8.0).

1.2 Scalability

By choosing which plugins are needed for a specific application Julius can be configured to run on low-end PCs to high performance parallel computers. We have run the software on PCs ranging from 350Mhz and 128MB RAM up to dual processor machines with 2.0GHz as well as Silicon Graphics workstations O2, Octane and a 4 Processor Onyx high performance computer.

1.3 Extensibility

Developers can add functionality to Julius by implementing their own plugins and registering them to the framework. This can be done at run-time without recompilation of the source code.

2 Methodology

The main part of Julius is the object oriented *JCore* library which encompasses the object communication system *(JOCS)* the Julius graphical user interface *(JGUI)* and the software development kit *(JSDK)* (fig. 2).

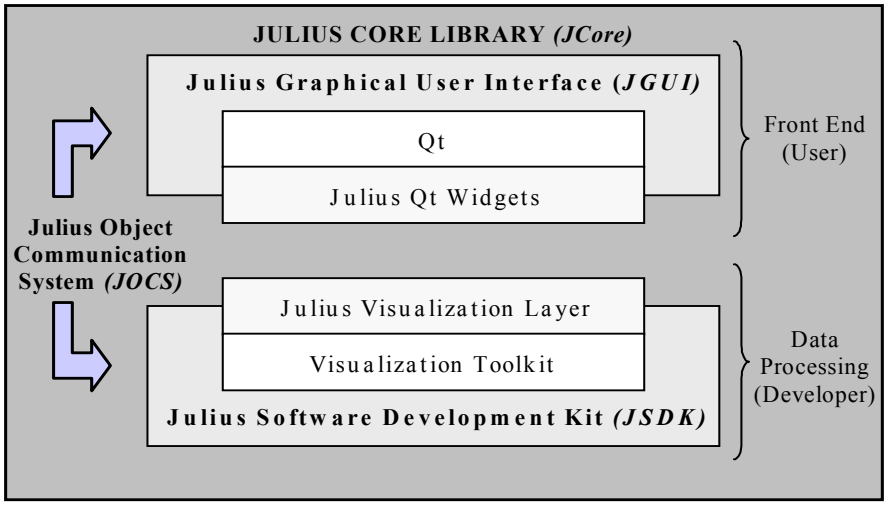


Figure 2. Conceptual overview of the framework.

2.1 Core Functionality

2.1.1 JOCS
The Julius object communication system *(JOCS)* is a message based information passing system between distributed objects in the framework. Senders can attach data to messages and select different addressing modes (single, multicast or broadcast mode) to minimize the traffic handling for the listeners. It also features posting of safe communication in multi-threaded environments. Custom message types are easily implemented by deriving from standard Julius message classes.

2.1.2 Plugins
Handling of plugin componentsa is another central part. Every feature in Julius is realized as a separate plugin which can be of the following types:

1. Generic plugins mainly used for message processing.
2. Data plugins to represent various forms of data.
3. Controllers for filtering and manpulating data.
4. Import and export plugins for different formats.
5. Navigation Plugins for support of external navigation hardware.
6. Renderer plugins for visualization of specific data types.
7. Configuration Plugins for one-time initialization of other plugins and applications.
8. Stepcard plugins that are added to the GUI for customized user interaction.

It is possible to have multiple versions of the developers own plugins running at the same time. Each of them can be easily activated and deactivated using the plugin configurator tool of the GUI which facilitates testing and evaluation of new components.

2.2 Extended Functionality

A number of plugins have already been developed for Julius. Data import formats supported are several Vista types, DICOM 3.0, Vtk, STL and RAW. From these sources visualization of surfaces, volumes, meshes, tensors and vector fields is supported. A transfer function editor allows for the editing of color and opacity values of the individual datasets. Additionally several datasets can be superimposed on one another and examined with a slice view which is visible in 2D and 3D. The filtering algorithms implemented include histogram analysis, threshold segmentation, manual editing, morphological filters, regiongrowing, marching cubes isosurface extraction, surface smoothing and decimation. Support for the Flashpoint and Polaris tracking systems has been included. To establish a sequence of events (e.g. filtering - marching cubes – smoothing - decimation, …) a workflow plugin has been implemented.

2.3 System Requirements and Dependencies

The software runs on any type of hardware. From low end PCs and laptops to high end workstations and HPCs. Mid to high end graphics hardware is recommended but not required. The framework uses Qt version 3.1 or higher and Vtk version 4.0 or higher. Julius has been tested on Unix, Irix, Linux and Windows operating systems.

3 SimBio related Plugins

3.1 Vista Import

The first plugin implemented was the import of the Vista file format into Julius (fig. 3.). The types of Vista datasets which are supported are volumes, surfaces, meshes, vector and tensor data. Vista data is converted internally to Vtk-derived formats to conform with the specifications of existing renderers for displaying volumes or polygonal data.

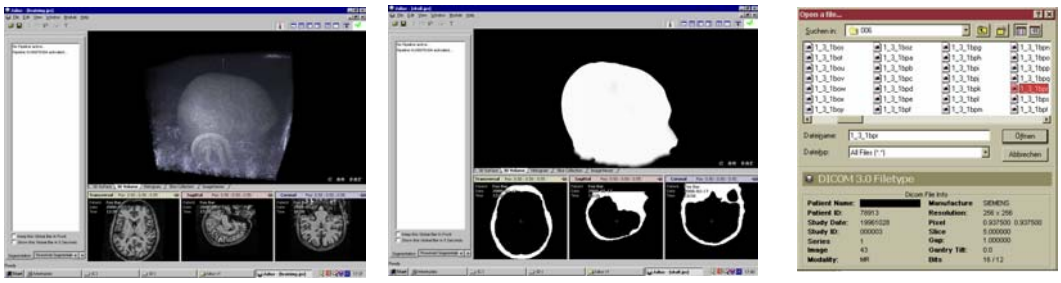


Figure 3. Visualized Vista files and extended file dialog for Vista Import

3.2 Stereo Plugin

The plugin for stereo rendering is capable of taking all datatypes which can be converted into Vtk-format and rendering them in a 3D mode using stereo glasses (fig. 4). This is a useful plugin for getting a sense of location on disconnected structures in 3D datasets. It can also be used for manual inspection of mesh quality during preprocessing for finite element analysis.

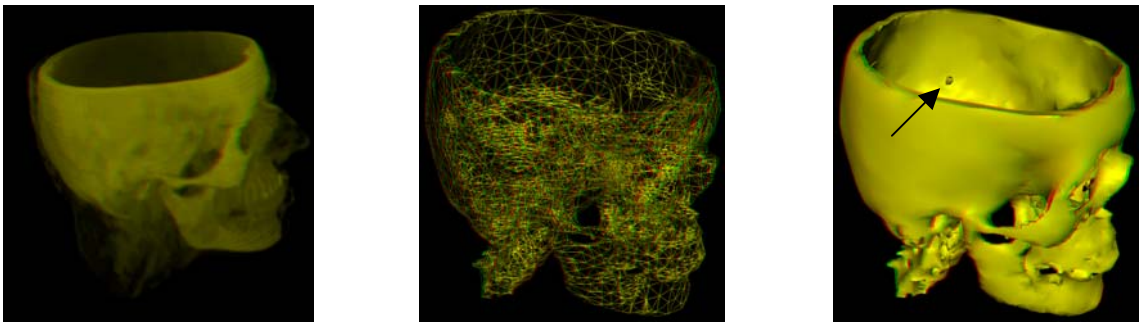


Figure 4. Stereo rendering of volume, mesh and surface data. In the right dataset location of the tumor (arrow) is easier to interpret than using mono rendering

3.3 Flythrough Plugin

The flythrough plugin is capable of generating animations from a scripted sequence of camera waypoints (fig. 5). The animation can be calculated and visualized on the fly. Output is to standard video format or individual bmp files for later conversion to video streams.

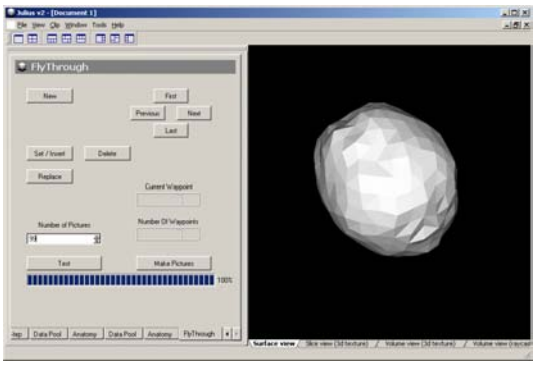


Figure 5: Flythrough generator controls for setting, adding, inserting and deleting waypoints and setting the number of generated pictures

3.4 Overlay Planes, Slice Views and Cut Planes

Overlays can be used to visualize two or more datasets at the same time in the slice view diagram. Currently several polymesh datasets can be shown simultaneously in the 3D view and combinations of any types in the slice view. Cut planes allow for the superposition of volume and slice view information (fig. 6).

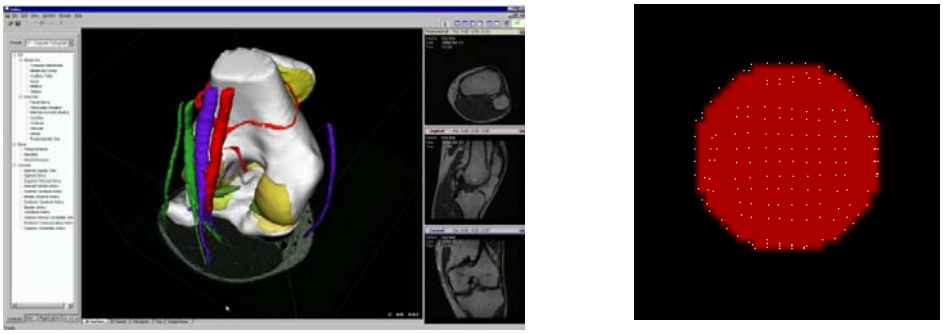


Figure 6: Left: Simultaneous display of 20 surface structures in the knee with cut plane view of slice information. Right: Overlay of example volume and mesh data in the slice view.

3.5 Direction, Interaction , Reaction Plugin (DIR)

A generic plugin for handling positional data from external navigation and tracking devices has been implemented. It allows for easy integration of any type of hardware for user input. To date we have used it to connect the computer mouse, Flashpoint[6] and Polaris[7] tracking systems as well as Phantom haptic devices[8]. Integrating a new device is done by implementing a thin layer that translates the coordinates received from the input device to the internal coordinate transformation format and passing that information to the DIR plugin. The DIR plugin gathers transformations from all devices, processes and combines them where necessary and applies the result to the relevant datasets. Devices can be configured to generate absolute or relative transformations establishing which systems transformations should take precedence over others. The plugin allows for multiple devices to interact with one or multiple objects within a scene or several objects being bound to different controlling hardware.

3.6 Phantom Plugin

Using the DIR plugin the Phantom haptic device has been integrated into the Julius software framework. The 6 degrees of freedom provided by the device enable an intuitive way to navigate within three-dimensional data, without cumbersome meta-keys required by two-dimensional input devices like the computer mouse. Users can simultaneously control camera position and angle.

Figure 7: Phantom haptic devices with 6 degrees of freedom and 3 (left) and 6 (right) degrees of force feedback.

4. Conclusion

The Julius software framework and its plugins provide a tool for visualization, manipulation and processing of medical datasets of various types. Its extensibility and adaptability allow the user to tailor and modify the systems capabilities to suit his/her context. With the plugins developed within the SimBio context the user can intuitively manipulate Vista datasets, have a better visual representation of spatial relationships and produce multimedia files for demonstration purposes.

References

[1] Keeve, E., Jansen T., Krol Z., Ritter L., von Rymon-Lipinski B., Sader R., Zeilhofer H.-F., Zerfass P., "JULIUS - An Extendable Software Framework for Surgical Planning and Image-Guided Navigation" Fourth International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI'01, Utrecht, October 14-17, 2001.
[2] von Rymon-Lipinski B., Jansen T., Krol Z., Hanssen N., Ritter L., Keeve E.,"An Extendable Application Framework for Medical Visualization and Surgical Planning" Proceedings Computer Assisted Radiology and Surgery CARS'01, Berlin, Germany, June 27-30, 2001.
[3] Jansen T., Rymon-Lipinski B., Krol Z., Ritter L., Keeve E., "An Extendable Application Framework for Medical Visualization and Surgical Planning," Proceedings SPIE Medical Imaging MI'01, San Diego, CA, February 17-23, 2001.
[4] www.kitware.com
[5] www.trolltech.com
[6] www.imageguided.com
[7] www.ndigital.com
[8] www.sensable.com