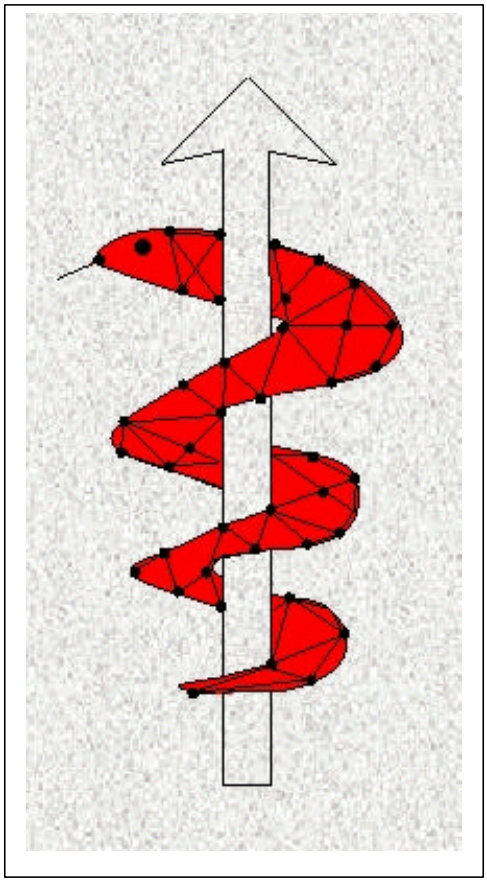
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D5a
Visualisation Design Report

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

Release History

Version	Date	
0.1	27.09.00	Initial Draft
1.0	25.09.00	

The SimBio Consortium :
NEC Europe Ltd. – UK
A.N.T. Software – The Netherlands
K.U. Leuven R&D – Belgium
ESI Group – France
Smith & Nephew - UK
MPI of Cognitive Neuroscience – Germany
Biomagnetisches Zentrum Jena – Germany
CNRS-DR18 – France
Sheffield University – UK

The SimBio Visualisation Module Requirements Specification

Markus Svensén
Max-Planck Institute of Cognitive Neuroscience

Frithjof Kruggel
Max-Planck Institute of Cognitive Neuroscience

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

This document is the requirements specification for the visualisation module (VM) of the SimBio project [SimBio00]. The purpose of this module is, as its name suggests, to visualise the results from other SimBio modules, e.g. stress fields from bio-mechanical simulations or source localisation and influence from bio-electrical simulations.

The purpose of this document is to specify the intended functionality of the VM and how this functionality will be available to a user through a graphical user interface.

1.1. The SimBio Project

The central objective of the SimBio project is the improvement of clinical and medical practices by the use of numerical simulation for bio-medical problems. The project builds on significant expertise and prior developments for specific applications to construct a generic environment, running on parallel and distributed computing systems, capable of handling a range of important problems relevant to the target community of clinical and medical service providers. This innovative development is an enabling technology for advanced clinical practice and health care leading to improvements in: non-invasive prognosis and diagnosis, pre-operative planning, design and implantation of prostheses and postoperative verification and evaluation of treatment success. The application evaluations in the project will demonstrate the effectiveness of the SimBio environment and thus accelerate the take-up of this IT technology within the medical area.

1.2. The Visualisation Module

The simulation of bio-medical data often requires advanced visualisation tools, capable of particularly accurate or high resolution visualisation. The SimBio VM will be designed to meet such requirement. It should be able to visualise simulation results with fast change of viewing parameters. It will allow the visualisation of conductivity and diffusion tensors as well as vector fields and iso-lines interpolated in force and electro-magnetic potential fields. Changes of time-dependent variables

are visualised with high update rates. The render engine will take advantage of the underlying hardware architecture, especially of a parallel processing environment and/or special purpose graphical display boards.

Export filters to standard animation and graphic formats will be implemented, to make data available to the cosmopolitan medical, clinical and engineering communities for presentation, publication and further investigation with external software tools.

Chapter 2. Requirements

This chapter describes the requirements to be met by the SimBio VM. The following, main section of the chapter describes the functional requirements. Non-functional requirements—user interface and performance issues—are discussed in the shorter final section.

2.1. Functional Requirements

The following sections describe the kinds of data objects that are recognised by the VM and how they can be visualised, covering the key functionality of the VM. The chapter ends with a short section on non-functional requirements, covering user interface and performance.

2.1.1. Data Objects

All the data sets occurring in the SimBio environment are defined over some finite grid in a 3D Euclidean space. There are two basic representations for the data sets: images (voxel volumes) and geometrical models (graphs). For image data sets, the grid is regular and rectangular, and the location of the nodes is only implicitly defined. For geometrical data sets, the grid is normally not regular and hence node positions must be explicitly defined.

For image data sets, the nodes always represent volume elements. Thus, there is an implicit second grid, in which the nodes are the points (vertices) that defines the corners of the volume elements. For geometrical data sets, both grids must be made explicit: in the vertex grid the nodes represent points, which are typically corners of volume or surface elements, which form the nodes of the second grid.

With the different kinds of nodes—points, surfaces and volumes—data may be associated. The data can take the form of scalars, 2-element complex numbers, 3-elements vectors or 6-element tensors. For each of these four forms, a number of visualisation modes are available, depending on the context in which the data appear (image or geometrical model; volume, surface or vertex). The data will have associated labels (e.g. intensity, rgb, potential, . . .), from which it may be possible for the VM to deduce a preferred visualisation mode in the given context. A

label-data tuple will be referred to as an *attribute* [1]. Nodes in a geometrical model may carry several attributes, some of which may visualised simultaneously.

2.1.1.1. Recognised Attributes

The VM will recognise the following attributes.

Table 2-1. Recognised attributes

Label	Value
gradient	vector(3)
intensity	scalar
rgb	vector(3)
complex	vector(2)
scalar	scalar
vector3	vector(3)
tensor6	vector(6)
potential	scalar
electrode	scalar, vector(3) (optional)
dipole	vector(3)
force	vector(3)
curvature	vector(3)
streamline	scalar

The attributes dipole, electrode and streamline are only recognised in geometric models.

New attributes may be defined; the visualisation of such new attributes would be selectable from the set of recognised with the same kind of value.

2.1.1.2. Data Object with Temporal Dynamics

Data objects may change over time, in terms of changing attribute values and, in the case of geometrical objects, changing vertex positions. Temporal dynamics will be

represented by repeated instances of attribute values and/or position, each instance holding the corresponding value at a single time step. 2D image objects with temporally changing attribute values will simply be represented by a concatenation of frames, each frame corresponding to one time step, containing the corresponding scalar or vector value. Temporally changing 3D image objects are not supported. Geometrical objects whose geometry, and maybe also attribute values, change over time will be represented by a sequence of distinct instances of the object. Temporally changing attribute values associated with a temporally constant geometrical model will be represented by repeated instances of the corresponding node attribute values, in a fashion similar to images with multiple frames.

2.1.2. Visualisation

Data objects can be visualised in several ways, depending on the kind of data and the overall data set representation. One visualisation instance (i.e. the display of a 2D slice from a 3D voxel volume or the rendering of a geometric object) is called a view.

Each view will have an associated 3D cursor position.

The VM will offer linking of different views, such that the 3D cursor position in one view can be mutually coupled to the 3D positions of other views.

2.1.2.1. Voxel Volumes

The VM will be able to display 2D slices of data (images) extracted from 3D, cubic voxel volumes along any of the three principal axes (centred normals of the sides of the cubic volume).

Scalar data will be displayed as pixel images using colour maps selectable for each view.

The VM will be capable of scaling scalar data to employ the full range of the available colour map.

RGB data will be displayed as pixel images with the corresponding colours.

The VM will offer the possibility to zoom 2D images.

The VM will offer mutual coupling (linking) of the zooming of any 2D image view with the zooming of other 2D image views.

The VM will offer the possibility to adjust contrast and brightness of 2D slices of scalar data individually for each view.

Scalar representation of multivariate data, such as magnitude of complex numbers and 3-element vectors and volume of 6-element tensors, will be displayed just like scalar values.

Complex numbers will be visualised by representing the real and imaginary components by different orthogonal colours (RGB or CMY). This may be combined with intensity indicating the magnitude.

It will be possible to display 3-element vectors by projecting them on the image plane. The projections may be directional or undirectional. This may be combined with colour or intensity indicating the magnitude. The vectors may be subjected to a global re-scaling.

It will be possible to display 6-element tensors by projecting the corresponding ellipsoidal representations (see 6-element tensors in geometrical models) on the image plane. This may be combined with colour or intensity indicating the volume. The tensors may be subjected to a global re-scaling.

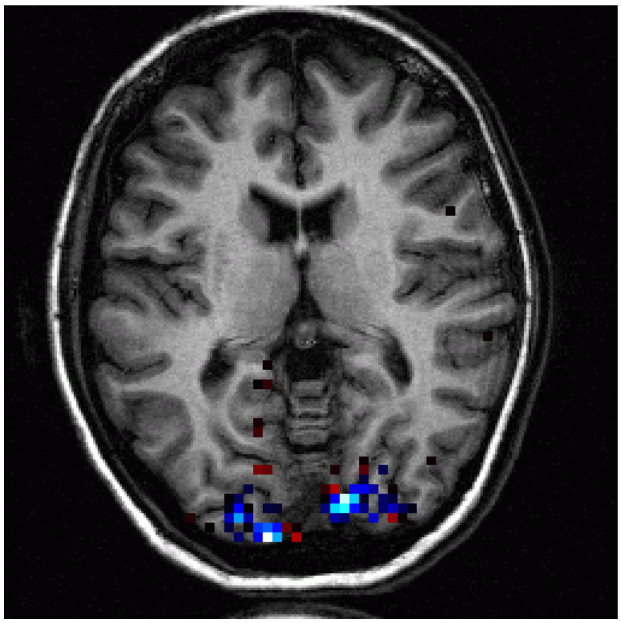
It will be possible to display the anisotropy of tensors by representing the three main cases (isotropic, planar and linear) by different orthogonal colours (RGB or CMY). This may be combined with intensity indicating the volume.

Image scalar data (explicit or derived, e.g. magnitude) may be used to define a display mask, excluding voxels above or below a given limit from visualisation. Alternatively, voxels falling out of range may be displayed with a for this purpose specified colour.

Images of scalar (explicit or derived) data, RGB data, and colour representations of complex and tensor data, constitute *pixel images*.

A pixel image may be overlayed on another pixel image, provided that the number of rows and columns of one of the images are integer multiples of the numbers of rows and columns of the other image, to yield a pixel image. An example is shown in figure Figure 2-1.

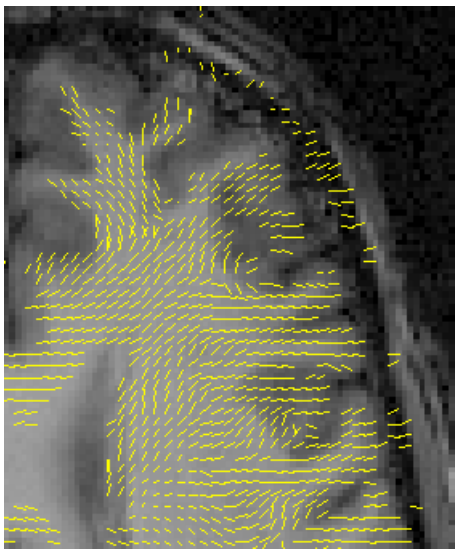
Figure 2-1. Greyscale image with masked overlay image



Pixel image overlays may be either opaque or transparent.

Images representing 2D projections of vectors or tensors may be overlayed on pixel images, yielding compound images. An example is shown in figure Figure 2-2.

Figure 2-2. Greyscale image (cropped) with masked overlay vector field



The default visualisation for whole images of scalar values is a scaled greyscale map; for overlay images a scaled non-greyscale colour map is the default.

The default visualisation for images of 3-element vector values is the corresponding 2D directional projection in the image plane, after a global scaling, with colour coding indicating the magnitude.

The default visualisation for images of 6-element vector values is the 2D projection of the corresponding ellipsoidal representation in the image plane, after a global scaling, with colour coding indicating the volume.

2.1.2.2. Geometrical Models

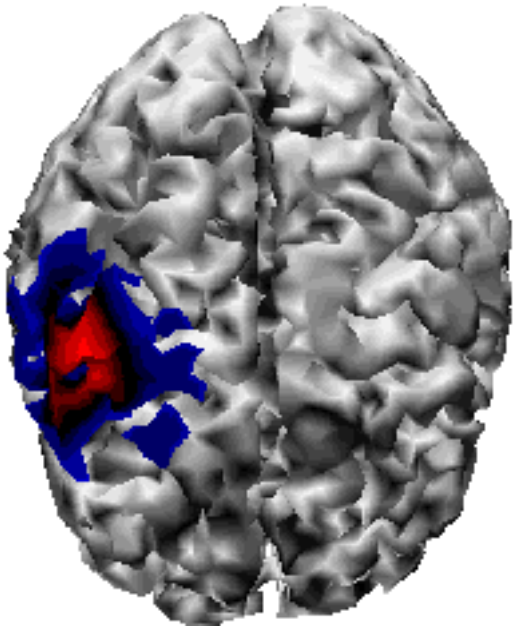
The VM will be able to render 3D geometric objects—meshes, surfaces and volumes—with variable rotation and scale.

Scalar and RGB values associated with volume elements will be displayed through the colour of the enclosing surfaces, in the scalar case using colour maps selectable for each view.

Scalar and RGB values associated with surface elements (polygons) will be displayed through the colour of the surfaces, in the scalar case using colour maps selectable for each view. Colour values associated with a surface override colour values associated with enclosed volumes.

Scalar and RGB values associated with vertices will be displayed through the colour of vertices and, where applicable, edge and surface elements. The colour of the latter will be interpolated between the colours of the vertices defining the surface element, in the scalar case using colour maps selectable for each view. Colour values associated with a vertex override colour values associated with enclosed surfaces or volumes. An example is shown in figure Figure 2-3.

Figure 2-3. A scalar field shown on a brain surface



The VM will be capable of scaling scalar data to employ the full range of the chosen colour map.

Scalar values associated with vertices may alternatively be displayed as iso-lines over the surface defined by the vertices.

Scalar representation of multivariate data, such as magnitude of complex number and 3-element vectors and volume of 6-element tensors, will be displayed just like scalar values.

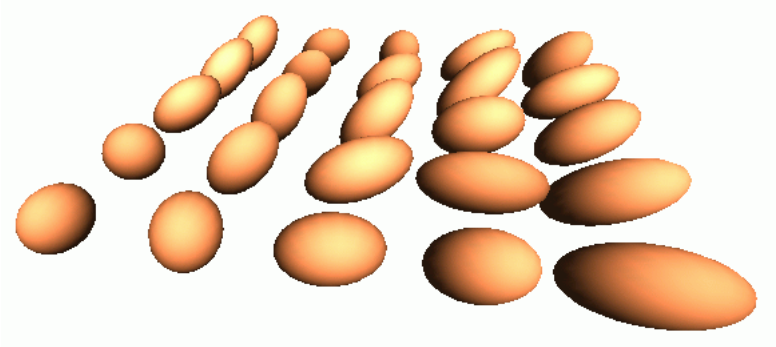
Complex numbers associated with volume elements, surface elements and vertices will be visualised by representing the real and imaginary components by different orthogonal colours (RGB or CMY), specifying the colour of the model surface according to the scheme governing scalar values. This may be combined with intensity indicating the magnitude.

3-element vectors will be displayed directly at the corresponding position (vertex position or surface or volume centre) in the 3D space. They may be directional or undirectional. This may be combined with colour or intensity indicating the magnitude. The vectors may be subjected to a global re-scaling.

6-element tensors will be displayed directly at the corresponding position (see prev.

para.) in the 3D space, using a 3D ellipsoidal representations, so called smarties. This may be combined with colour or intensity indicating the volume. The tensors may be subjected to a global re-scaling. An example is shown in figure Figure 2-4.

Figure 2-4. Visualisation of tensors using ‘smarties’



Anisotropy of tensors associated with volume elements, surface elements and vertices will be visualised by representing the three main cases (isotropic, planar and linear) by different orthogonal colours (RGB or CMY), specifying the colour of the model surface according to the scheme governing scalar values. This may be combined with intensity indicating the volume.

Scalar data (explicit or derived, e.g. magnitude) may be used to define an element mask, such that elements falling out of range may be displayed with a for this purpose specified colour. An example is shown in figure Figure 2-3.

The VM will offer the use of clipping planes for excluding parts of a geometrical dataset from being visualised.

Several geometrical models may be displayed in a single view.

2.1.3. Special Data Objects

Following data objects will be subject to special treatment by the VM:

- electrode,
- dipole and

- streamline.

electrodes, which may have an empty value field, will be plotted in 3D using an electrode glyph; alternatively, they may have a scalar value field, which will be represented by the colour coding of the glyph. The may also have a 3-element normal vector giving the orientation of the glyph.

dipoles will be plotted as 3D vectors, centred at the corresponding vertex position, with a positive and negative pole (half) coloured red and blue, respectively.

streamlines are assumed to consists of a connected set of vertices, possibly with an associated scalar attribute representing the strength of the streamline. The will be visualised as a set of connected ‘tubes’ along the edges connecting the vertices, where the strengths is indicated by the thickness of the tubes.

2.1.4. Generation of Animations

The VM will offer the possibility to generate animations that may be transformed into common animation formats (see Section 2.1.5.2).

Animations may be generated from data objects with a temporal dimension, where each time step in the data will be shown in one or more frames in the animation, as selected by the user.

Animations maybe generated from a sequence of changes in the viewing parameters, controlling the view of a geometric object. An example would be an animation with a rotating brain.

2.1.5. Other Functional Requirements

In addition to the key functionality defined in the previous sections, the VM will deliver functionality allowing the user to conveniently specify input data for other SimBio routines. It will also offer the possibility to export visualised results in at least one common graphic format.

2.1.5.1. User Interaction

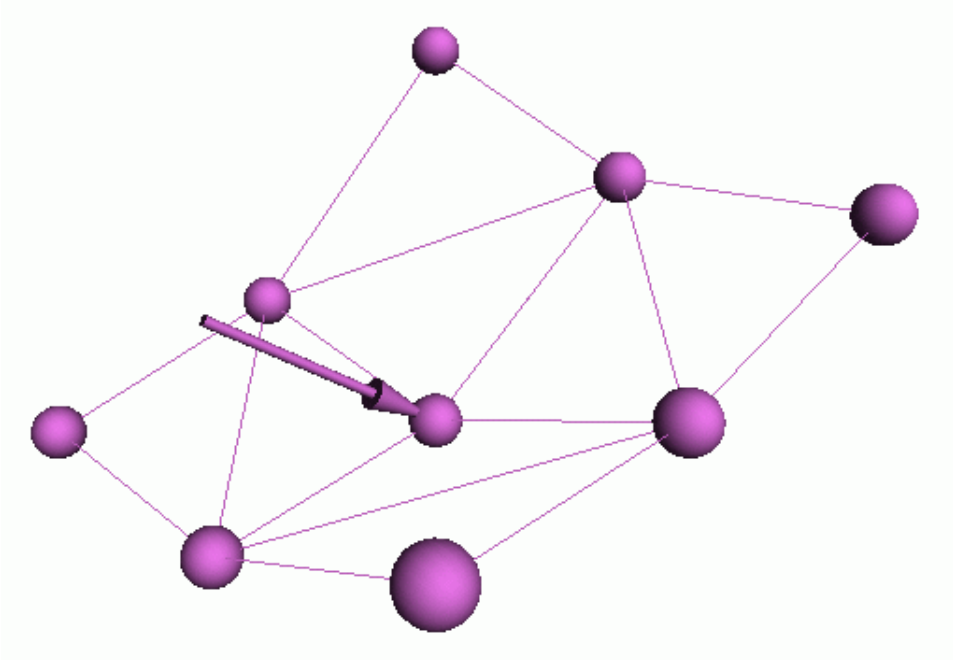
The user will be able to interactively probe the value (scalar or vector valued)

associated with any grid position in a SimBio data set.

The VM will offer the user the possibility to associate vectors representing mechanical forces or electric dipoles, with vertices in a given data set and save these together with the data set. An example is shown in figure Figure 2-5.

The VM will offer the user the possibility to ’tag’ vertices in a given data set and save tags these together with the data set. This may for example be used to mark fixed boundary vertices for simulations.

Figure 2-5. Representation of a force entered by the user



2.1.5.2. Export Formats

The VM will support export of visualisation results in common graphics and animation formats.

The VM will be to export images in formats employing

- lossless or no compression (e.g. PNG, TIFF and PNM), as well as

- ‘lossy’ compression (e.g. JPEG).

Several tools exist, freely available on the Internet, for converting between these formats, as well as into other formats. These tools may be distributed together with the SimBio VM.

The VM will not support direct generation of animations. However, it will support batch export of sets of images corresponding to a temporally changing data object, into a set of graphic files in one of previously listed graphic export formats. Again, several freely available tools exist for compiling such sets of graphic files into a single animation.

The SimBio VM will offer an integrated interface to the tool ImageMagick™, if available, for export of visualisation results.

2.2. Non-functional Requirements

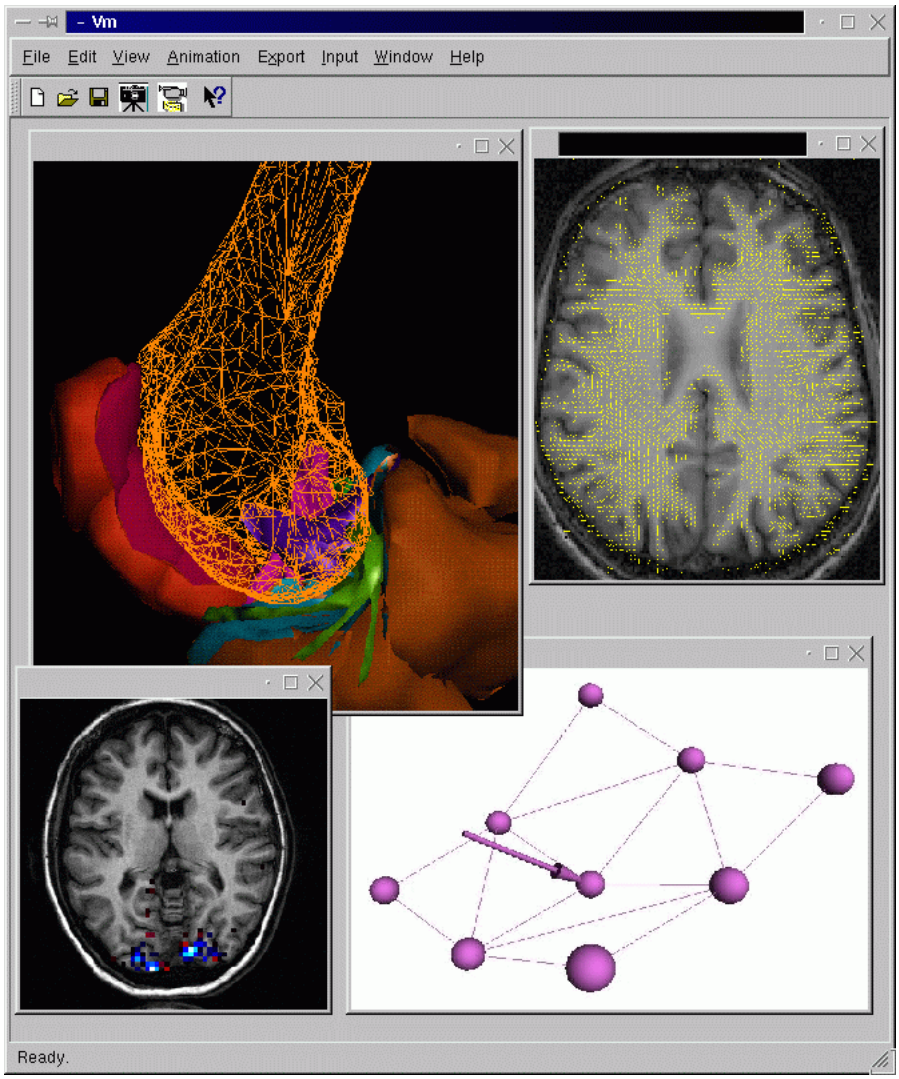
2.2.1. User Interface

The VM will offer a graphical, window based user interface, controlled by the user through keyboard and mouse. Common commands, such as selection of data sets for visualisation, will be available through menus and toolbar buttons; other commands will be available through menus. An illustration of what this could look like is shown in figure Figure 2-6. In addition, a simple script language interface will be included for control of complex off-line operation (e.g. generation of animations) and launching of the VM from other modules.

Commands that involve direct interaction with the data, such as zooming a region of an image or rotating a geometrical data set (e.g. a geometrical model of the brain surface) on the screen, will be accessible through mouse actions. For example, clicking with a mouse button on a particular location on an image will zoom the image around the selected location. Similarly, grabbing (pressing a mouse button over) a particular location on a display area of a geometrical data set and dragging it (moving the mouse while keeping the mouse button pressed) will result in a corresponding rotation of the geometrical model (or a simplified geometrical

representation thereof).

Figure 2-6. The graphical user interface



2.2.2. Performance

Some of the VM's tasks can be computationally demanding, e.g. rendering of a geometrical model with hundreds of thousands of polygons. The computational requirements are typically depending on the quality of the visualisation. In addition,

hardware resources, which vary from platform to platform, may impact what kind of strategies that can be adopted to achieve efficient visualisation. Therefore, the VM should offer the user the possibility to control the trade-off between quality and speed of visualisation.

Notes

1. The notion of an “attribute”, as used here, is not necessarily identical with the attributes of data objects stored in the Vista file format [Kruggel00], although in many cases there will be a one-to-one correspondence.

Chapter 3. Realizing the Requirements

The VM will be built using the Qt™ [Qt], OpenGL® [OpenGL] and Vista [Vista] libraries under X-windows™ and Linux®. Qt is a library for building graphical user interfaces (GUIs) that provides elements such as windows, menus, dialog boxes, etc. OpenGL is a graphics library designed for efficient rendering of 3D geometrical objects, which is also capable of displaying pixel images.

The use of Qt and OpenGL should make porting to Microsoft® Windows® 95/98/NT straightforward. However, such a port, which at the moment would require a commercial licence for the Qt library, will not be considered further within the SimBio project.

The OpenGL library is believed to offer all the low-level graphics functionality that will be required to create the advanced visualisation routines required in the SimBio project. A specific extension for the Qt library makes the integration of the functionality provided by OpenGL into a GUI built using Qt straightforward.

The signal-slot mechanism provided by the Qt library is a well suited infrastructure for linking properties, such as 3D position and zoom, between different views.

The Vista library offers efficient data structures and associated algorithms for creating and working with the data objects discussed in section Section 2.1.1.

Although the VM will be implemented under Linux, it will also ported to and tested under other UNIX® dialects, such as Irix®.

Bibliography

[SimBio00] The SimBio Consortium (2000), *SimBio: A generic environment for bio-numerical simulation*, `http://www.simbio.de/`

[Kruggel00] F. Kruggel and M. Svensén (2000), *File Format Conventions for the SimBio project*, `http://www.simbio.de/`

[Vista] The Vista web-site: `http://www.cs.ubc.ca/nest/lci/vista/vista.html`

[OpenGL] The OpenGL web-site: `http://www.opengl.org/`

[Qt] The Qt web-site: `http://www.trolltech.com/products/qt/`