map3d User's Guide

Version 7.2

Last update: June, 2017

Authors: Rob MacLeod (macleod@cvrti.utah.edu) and
Bryan Worthen (worthen@sci.utah.edu)

Scientific and Computing Institute (SCI)
www.sci.utah.edu

Center for Integrative Biomedical Computing (CIBC) www.sci.utah.edu/cibc

Nora Eccles Harrison
Cardiovascular Research and Training Institute (CVRTI)
www.cvrti.utah.edu
the University of Utah

Support for this project came from:

The NIH National Center for Research Resources (NCRR) and the Nora Eccles Treadwell Foundation

Contents

1 What's New?

In this section, we highlight the latest additions to *map3d* in the (vain?) hope that people will read at least this much of the manual and be able to quickly make use of the latest and greatest that the program offers.

1.1 Version 7.2

This is the seventh version of the "new" *map3d*, the latest with a Qt-based GUI.

Some of the specific additions that you should notice over previous versions include:

GUI improvements: *map3d* 7.0 was rewritten in Qt. This was largely to help cross-platform issues that were arising and the deprecation of some features being relied on. Most main windows are preserved, but some GUI dialogs have received a face lift.

old format deprecation: *map3d* has deprecated the old .geom, .tsdf, .data, and .tsdfc data files, in favor of matlab.

0-based or 1-based indiced: .fac or .mat geometries can refer to points in a 0-based or 1-based manner.

Time-dependent geometry: *map3d* 7.2.2 has introduced time-dependent geometry. See Section 7.3.1

64-bit Windows support: *map3d* 7.2.4 has introduced 64-bit binaries for Windows, enabling the use of large datasets.

Bug fixes: not that the previous version had any bugs, but we found a few(!) more things to fix.

1.2 In the works ("vapourware")

A small sampler of things that are in the works:

- Saving the frames into actual movies.
- Dynamic menus that indicate the current parameter selections (a bit more in progress).
- New display modes for the display of vectors.
- Fixing the bugs listed on the bugs page (see Section 2.3).

2 Introduction

This document describes the function and usage of version ṽersion of the program *map3d*, a scientific visualization application originally developed at the Nora Eccles Harrison Cardiovascular Research and Training (CVRTI) and now under continued development and maintenance at the Center for Integrative Biomedical Computing (CIBC), which is an NIH/NCRR funded center hosted by the Scientific Computing and Imaging Institute (SCI) at the University of Utah. The original purpose of the program was to interactively view scalar fields of electric potentials from measurements and simulations in cardiac electrophysiology. Its present utility is much broader but continues to focus on viewing three-dimensional distributions of scalar values associated with an underlying geometry consisting of node points joined into surface or volume meshes.

map3d has been the topic of some papers[10–13] and a technical report[9] and we'd love it if you would reference at least one of them (perhaps[13] or[10] are the easiest ones to get copies of) as well as this manual when you publish results using it. There have been many many more papers that use *map3d* and the list keeps growing.[1–10, 13–31]

As with our past versions, this edition of *map3d* is completely open source. Anyone can download not only the executable for some major platforms (OSX, Windoze) but also the complete source code for the program. Please note that we do not have a good way yet to incorporate changes people outside our little group make to the program. If you do wish to change and then contribute back, please let us know as soon as possible and we can try and coordinate as best we can. Of special interest are any ports of *map3d* to another platform—please let us know about this and we can add it to the list and release it with the rest.

2.1 Acknowledgments

The history of *map3d* goes back to 1990 and the first few hundred lines of code were the product of a few hours work by Mike Matheson, an inspired visualization specialist, now with SGI in Salt Lake City. This was my (Rob) introduction to GL and C and this program became my personal sand box to play in. Along the way, Phil Ershler made valuable contributions in figuring out the magic of Formslib for some user interface controls and developing with me *graphicsio*, the geometry and data file library that supports *map3d*. Ted Dustman has recently taken up maintenance and extensions of *graphicsio* and remains my main man when I need programming lessons.

This is one in a series of "new" versions of *map3d*, the series (labeled 5.x or above) that marks the move from GL to OpenGL library and thus to becoming truly portable. In fact, we call the old one *map3dGL* now to indicate its links to SGI's original GL library. We seem permanently stuck in the middle of this big conversion project, moving support to OpenGL and adding lots of power as we convert functionality. The reason for the version 6.x, was the move to gtk as the GUI library with which we create all the dialog and display elements of the program. This move has allowed us to extend dramatically the set of dialog boxes *map3d* offers. Version 7.x moved again to the Qt toolkit, preserving the spirit of the dialogs of the Gtk+ version, and this newest version 7.2 contains many examples.

There are some people who have been instrumental in the process and deserve special mention. Chris Moulding is a graphics programmer and general software whiz who surveyed my sand box architecture, pulled together the essential walls, created new ways to make rooms, and still left lots of the sand box around so we could continue to play. From version 5.2 onward, Bryan Worthen replaced Chris and really has found the spirit of *map3d*. Bryan has become the main driving force behind the actual work of coding and fixing. He strayed off to some other project for a while, but never lost his love for *map3d*; we are really pleased that he has returned to pick up the torch again. J.R. Blackham joined the team in about 2003 while finishing an undergraduate in Computer Science at Utah and a little after. Jeroen Stinstra is my super-postdoc, helpful in more ways than I knew I even needed and full of inventive ideas. He has created the support for MATLAB that we use in *map3d* (and the SCIRun project) and is best bug-catcher I know.

The largest thanks must go to the users of *map3d*, who provided the real inspiration and identified the needs and opportunities of such a program. Among the most supportive and helpful are Bruno Taccardi, Bonnie Punske, and Bob Lux, all colleagues of mine at the CVRTI. Dana Brooks and his students from Northeastern University are also regular users who have provided many suggestions and great enthusiasm. Also invaluable in the constant improvement of the program are my post docs, Jeroen Stinstra, and graduate student Quan Ni, Rich Kuenzler, Bulent Yilmaz, Bruce Hopenfeld, Shibaji Shome, Lucas Lorenzo, Andrew

Shafer, and Zoar Englemann. They give me new energy every day and remind me why I am a professor. Notable new additions to the family are Randy Thomas from Universite d'Evry Val d'Essonne in Evry, France. The great thing about Randy is that he used *map3d* to visualize concentrations of ions in his simulation of the nephron! Also, Ed Ciaccio from Columbia University has become a big user and even takes it to his classes.

The first user and long-time collaborator and friend was Chris Johnson and this new version of *map3d* is possible because of the success he and I have had in creating the SCI Institute and specifically the NIH/NCRR Center for Integrative Biomedical Computing (www.sci.utah.edu/cibc).

We gratefully acknowledge the financial support that has come from the NIH, National Center for Research Resources (NCRR), the Nora Eccles Treadwell Foundation, and the University of Utah, which provides us with space and materials to create this sand box. The Nora Eccles Treadwell Foundation has provided support not only for the development of *map3d* but also the experiments that provide the huge pile of data we have used *map3d* to analyze.

Rob MacLeod, June, 2017.

2.1.1 Open Source License

The terms of the license agreement under which we release *map3d* are simple and as follows:

2.1.2 Libraries used by *map3d*

map3d incorporates the functionality of several external libraries. They are:

- Qt - Qt - Copyright (C) 2016 The Qt Company
- PNG - Copyright (c) 1998-2002 Glenn Randers-Pehrson
- Jpeglib - Copyright (C) 1991-1998, Thomas G. Lane.

We use Qt to interface with the window manager to give us windows with OpenGL capability, as well as giving us widgets we need for interactive control. We use PNG and JpegLib to be able to save `.png` and `.jpg` images of *map3d*. All of these libraries are covered by the GNU LGPL, which is included in the distribution of *map3d*.

As of version 7.2, we also release internal libraries under the same license as above for the rest of *map3d*.

2.2 Documentation

This document should have reached you either as a pdf file or via the *map3d* web site. If you would like more copies or the latest version, go to the same web site and look for the links under Documentation:

`www.sci.utah.edu/software/map3d.html`

2.3 Bug reporting

We want to hear your response to using *map3d* and especially to learn about any bugs you may find. They may be features, rather than bugs, but if so, we will be happy to hear your impressions.

To submit a bug report please send email to map3d@sci.utah.edu or point your browser at *map3d* issue tracker (you will need to register your e-mail address) with the following information:

1. Type and version of the operating system and hardware you are using.
2. Version of *map3d*.
3. Description of the bug/feature you encountered.
4. Suggestions for fixing the bug or altering the program behavior.

Here is a short list of the issues we know about and are currently addressing:

- When adding surfaces or updating time series data interactively, there can be surprises with scaling. The easy solution is to adjust the scaling in some small way and the resulting update usually takes care of the problem.

2.4 How to reach us

We have established an email address for *map3d*, map3d@sci.utah.edu, and web pages within the website www.sci.utah.edu/ncrr dedicated to *map3d*. There is also a majordomo mailing list for *map3d* users called `map3d-users@sci.utah.edu`. To subscribe to this list, send email to `majordomo@sci.utah.edu` with the following in the message body

```
subscribe map3d-users
```

Please let us know how you use *map3d* and how we can make it better for your purposes. We can only develop this program with continued support and the best way to achieve this is to show that others use the program and find it helpful.

3 Installation

3.1 System requirements

map3d is written in standard C/C++ and uses the OpenGL and Qt, both choices made to ensure broad portability of the program.

Note: As of *map3d* 7.1 or greater, Qt version 5.4.0 or greater is required to run *map3d*. Please update your dependencies according the the Windows, Linux, or Mac sections below.

All platforms: OpenGL now comes standard on most systems. Instructions on how to install Qt are described in detail below based on which platform you are installing. described below

Requirements for all systems.	
OpenGL libraries (GL and GLU)	version 1.1 + [1]
OpenGL/window interface library (GLX)	
GTK+ libraries and dependencies	version 2.4+
gtkglext: GL extension for GTK	1.2+

Linux (i386): we do not supply binaries for Linux, but *map3d* should be able to be built from source. It requires the OpenGL library, which is available as the mesa library at www.mesa3d.org for any Linux platform, and Qt 5.4 or greater. For better performance, graphics cards from companies such as nVidia (www.nvidia.com) usually provide OpenGL libraries.

Requirements	
Operating System	kernel 2.2.x
Architecture	x64
Applications Binary Interface	libc2.1
Recommendations	
Window system	XFree86 version 4.0 +
Hardware	3D graphics card (nVidia, 3dfx, ati)
	$\geq$ 128 MB main memory

Windows:

Requirements	
Operating System	XP/7/8/10
Architecture	x64
Recommendations	
Hardware	3D graphics card (nVidia, 3dfx, ati)
	$\geq$ 128 MB main memory

Mac OS X:

Requirements	
Operating System	Mac OS 10.8 and newer
Architecture	Intel
Recommendations	
Hardware	3D graphics card (nVidia, 3dfx, ati)
	Intel processor
	$\geq$ 512 MB main memory

SGI: we have dropped our support for SGI. This platform, while so central to the origins of *map3d* and OpenGL, is just not a significant desktop system in this day and age.

3.2 General Installation

With *map3d* moving to Qt for window support, Mac and Windows installers are available which should just install. It is still not as easy as past versions, which required just the download of an executable, but for now we will attempt to make installation as easy as we can. Simplified instructions will be in a README file which comes with each package, and are also listed below:

To download the software, use this URL `www.sci.utah.edu/software/map3d.html`, and click on the "Download" button. You'll need to sign into the SCI archive. For each of the installation instructions below, you can grab those file from this page.

To test the installation, use the test files that accompany this distribution. Each has some script files included that show how to call and execute *map3d*.

3.3 Linux Installation Instructions

Linux installations are not available at this time. See Section 3.6 for instructions on compiling and running *map3d*.

3.4 Windows Installation Instructions

The Windows installation is relatively straightforward. You just need to download and run the *map3d* installation package. *map3d* can be run from the Start menu or the Program Files directory.

3.5 Mac OS X Installation Instructions

The Mac OS X installation is relatively straightforward. You'll need to download *map3d*'s DMG file, and then install it. Inside the DMG is a file call *map3d*. This is the executable and can be run from there. All the dependencies are included in the DMG.

3.6 Installing from source

We have tried to make installing *map3d* from source as simple as possible. There are four steps:

1. Download the source
2. Download and install map3d's external dependencies
3. Setup the make configuration
4. Compile

3.6.1 Download Source Code

You can get the *map3d* source code from the *map3d* download page at `http://www.sci.utah.edu/software/map3d.html`. Login and work your way to the *map3d* version 7.2 download page, and download the source code.

3.6.2 Install Dependencies

You will need to install *map3d*'s Qt development environment for 5.4 or greater. For Windows, Linux, and Mac, this is usually pretty straightforward. The Qt archives are located at https://download.qt.io/archive/qt/5.4/5.4.0/

3.6.3 Configuring *map3d*

You should not have to change much in order to get *map3d* to compile. MatlabIO (one of *map3d*'s dependents) needs to know where ZLIB is (required, and should be installed after completing the previous section), and *map3d* needs to know where Qt is. The included files show samples of how this is to be done, and following are specific details.

Usually all that needs to be done is to invoke qmake and make. These procedures have been wrapped in a script for both Windows and Mac, see the README file at the root of *map3d*'s download page.

Windows is set up to use Visual Studio 2015 with Qt 5.4.0, both in 64-bit mode. Windows ZLIB binaries are included in the source distribution.

Remember for *map3d* to run, you should probably add *map3d* to your path. If, when you run *map3d* see errors like "Cannot load library Qt5Core.dll" or "Cannot map so libQt5Core.so.5", then you need to add the Qt libraries to your runtime path.

Contact If there are problems, feel free to contact map3d@sci.utah.edu

4 Usage

This version of *map3d* provides two ways to load files. The first is via the command line, which is described in this section. The second is via the files window (see Section 8.3.1). You can also launch *map3d* with some command line options and then modify the associated settings using interactive menus and the files window.

This is a subset of *map3d*'s usage:

```
map3d  -b -nw -nv
       -f geomfilename
         -w
         -as xmin xmax ymin ymax
         -al xmin xmax ymin ymax
         -at xmin xmax ymin ymax
         -t time-signal-number
         -c mesh colour
         -p scalar data (potentials) filename
           -s num1 num2
           -i increment
           -ph maxpotval
           -pl minpotval
           -cs contour-spacing
           -ps scaling_value
           -ch channels-filename
           -sl surfnum
           -ff fidfile
         -lm landmarks-filename
         -ll leadlinks-filename
         -lh
```

4.1 Typical usage examples

Here are some typical examples of using *map3d*:

- Display the contents of a geometry file:

  ```
  map3d
  map3d -f geomfilename.pts
  map3d -f geomfilename.fac
  map3d -f geomfilename.mat
  ```

  The first instance will run map3d and allow you to input files interactively (see Section 8.3.1).

  The first form with arguments reads only the node points (.pts file extension) while the second form also reads the connectivities from a .fac file and displays both mesh and nodes. The third form assumes that a matlab geometry file (.mat extension) exists that contains both nodes and connectivities. We describe all these forms of geometry files in Section 6.1.

- Map scalar values from a single time instant stored in a "pot" file (described in Section 6.3.1):

  ```
  map3d -f geomfilename.fac -p datafilename.pot
  ```

- When there is a mapping required between the potentials and the geometry, *e.g.,* when the order of values in the .pot and .pts file are not identical, we require a channels file (see Section 6.4 for details of the channels files),

```
map3d -f geomfilename.fac -p datafilename.pot -ch thefile.channels
```

- To display a *time series* of scalar values on the geometry, the basic format is the same

  ```
  map3d -f geomfilename.fac -p datafilename.mat
  ```

  with the time series stored in a datafile described in Section 6.3.

- Geometry can also be stored in a matlab file (in a format described in Section 6.1.3). The command format is essentially unchanged

  ```
  map3d -f geomfilename.mat -p datafilename.mat
  ```

  except that channel information is usually contained in the .geom file and thus seldom needs to be specified explicitly.

- A time series data matlab file (.mat) contains a sequence of potentials, as described in Section 6.3.2. To select a subset of the time series for display, append the parameters `-s` and, optionally, `-i`, for example,

  ```
  map3d -f geomfilename.fac -p datafilename.mat -s 1 100 -i 2
  ```

  to select time instants 1 to 100 with an increment between instants of 2 (*i.e.,* 1, 3, 5, 7, . . . , 99).

- Another way to describe a time series is through a series of .pot files that are numbered in sequence. For example to read a sequence of the files `mapdata001.pot`, `mapdata002.pot`, `mapdata003.pot`, . . . `mapdata009.pot`

  ```
  map3d -f geomfilename.fac -p mapdata -s 1 9
  ```

- Matlab geometry files can contain more than one geometry so that we need to select a specific collection of nodes and connectivities for the display, by means of an "@" suffix to the geometry filename specification. Calling

  ```
  map3d -f geomfilename.mat@2 -p datafilename.mat
  ```

  will select geometry #2 from the file `geomfilename.mat`.

- Multiple instances of `-f` create multiple surfaces, which by default all appear in the same window. Adding the `-nw` option creates a separate window for each of the surfaces. So a typical call would look like

  ```
  map3d -f geomfile1.fac -p thedata1 -f geomfile2.fac -p thedata2
  ```

  However, you can include all the regular features for each of the surfaces so that things can get much more complex. For multi-surface displays, it is often better to use script files (see Section 5) below.

This version of *map3d* provides an interactive means of specifying geometry numbers from a .geom file or time instants from a time series data file (see Section 8.3.1).

4.2 Global Parameters

The following general parameters affect the entire display:

`-b` = open each individual window without borders placed within a master window that still has the usual borders. To move or resize individual windows, hold the Alt (meta) key and use the left and middle mouse buttons, respectively. Most of these can be anywhere on the command line. Also, if you use `-b` without any other arguments, *map3d* will allow you to select the files interactively and add them to this master window.

`-v` = show current version of the program. If this is the only argument, the program will exit.

`-nw` = for multiple surfaces (*i.e.,* more than one set of points and triangles), place each surface data in a new window. By default, *map3d* opens a single window for all surfaces.

`-slw 0` = do not show any legend windows at startup.

`-fs interval` = Sets the run-time interval between frames as accessed by the arrow keys. Note that this is independent of the `-i` option below, which subsamples the data as it is being read in. This feature can be changed via the menus at runtime.

`-nv` = to NOT check validity of geometry files. This can have a large impact on startup performance if *map3d* needs to load large geometries.

`-c colour` = colour value to use on all surfaces for which there is no specific colour specification. This option must be set before any surfaces are read, since the same option sets the colors for individual surfaces. See the mesh-specific `-c colour` below for colour examples.

`-bg colour` = colour value to use as background of all windows for which there is no specific colour specification. This option must be set before any surfaces are read, since the same option sets the colors for individual windows.

`-fg colour` = colour value to use as text colour for all windows for which there is no specific colour specification. This option must be set before any surfaces are read, since the same option sets the colors for individual windows.

`-if basefilename` = base filename for any image files that are generated in this run of the program.

`-dp datafile_pathname` = directs the search for data files accessed to another directory. Using an alternate pathname, you can override the original directory specification for the files and get them from, say, an optical disk. This value can also be set with an environmental variable called `MAP3D_DATAPATH`, which you can set at any time before executing *map3d*. With this option, *map3d* looks in `datapath/filename`.

`-bgi image_filename` = An image file (in jpg or png) to display in the background. Used in conjunction with -bgp.

`-bgi x1 y1 z1 x2 y2 z2` = Coordinates in world space to display the image specified with -bgp. If unspecified, the image will fill the geometry window.

`-ss` = Sets the scale to be the same between all windows. Typically *map3d* will position each window such that its geometries fill up the window, but this will enforce that all positioning is done the same.

`-sr` = Sets the amount of time (sample rate) between frames, for display purposes in the time series window. This is on a time-unit basis; e.g., if the time unit is ns, and the sample rate is 3, the time series window will show that 3 ns passes with every frame advance. Works in conjunction with the *-tu* option.

`-tu` = Sets the time unit to display in the time series window. Works in conjunction with the *-sr* option.

`-ss` = Sets the scale to be the same between all windows. Typically *map3d* will position each window such that its geometries fill up the window, but this will enforce that all positioning is done the same.

4.3 Geometry specifications

The basis for display in *map3d* is one or more geometry descriptions, which are usually in the form of surfaces, but can also be a set of line segments or tetrahedra; hence we can picture each set of nodes and connectivities as a "meta-surface", which we generally refer to as a "surface". For each such surface, *map3d* needs the set of node locations in three-dimensional space and usually some connectivity information that defines the (meta) surface. The geometries must exist in discrete form and be stored in files that *map3d* can read (see Section 6.1 for details of the file formats). There is no provision at present for analytically defined geometries.

To tell *map3d* where to look for this geometry information, each occurrence of `-f` in the command line indicates that beginning of a new surface. All parameters (except for global options) that follow before the next occurrence of `-f` refer to the current surface.

`-f geometry-file` = filename of the geometry file(s) containing points and connectivity information. Legal formats for the file specification are:

1. nodes (.pts) file will read and display only the nodes from the geometry; no display of the potentials is possible with just this information;
2. triangles/tetrahedra (.fac/.tetra) file will read **both** the connectivities and the nodes (provided both exist and share the same root filename);
3. binary matlab geometry (.mat) file contains both nodes and connectivity information and may also contain channel mapping. At present, multi-surface geometry files must include a specific indication of the desired surface (`@surfnum`); otherwise, *map3d* reads all surfaces in the file.

Note: by specifying a root filename without any extension, *map3d* will look for all valid geometry files and try and construct the most comprehensive set. (It will do the same for data files as well.) Where there are multiple, potentially conflicting files with the same root, *e.g.,* `file.pts` and `file.geom`, *map3d* will select binary over text files. See Section 6.2 for more details on the rules for specifying and reading geometry files.

`-w` = place this and subsequent surfaces in a new window. This option will do nothing if the `-nw` option is set or if this is the first surface

`-fg colour` = desired colour for the screen information of a particular window, if this will be specified as a red, green, and blue value triplet ranging from 0 to 255. Some examples are:

255 0 0	red
0 255 0	green
0 0 255	blue
255 255 0	yellow
255 0 255	magenta
0 255 255	cyan
255 255 255	white

`-bg colour` = desired colour for the background of a particular window, specified as a red, green, and blue value triplet. See the `-fg option` for examples.

`-c colour` = desired colour for the mesh of a particular surface, specified as a red, green, and blue value triplet. See the `-fg option` for examples.

`-as xmin xmax ymin ymax` = set the absolute location in pixels of the surface window most recently defined. We assume an origin in the lower left corner of the screen and the typical full screen of an SGI workstation with a 19-inch monitor has 1280 by 1024 pixels. This option is useful for setting consistent layout of windows, especially when there are multiple surfaces, each in its own window.

`-al xmin xmax ymin ymax` = set the absolute location in pixels of the surface window most recently defined's colormap legend window. There will be one of these windows per surface only if a valid data file is associated with it.

`-slw 0` = do not show the legend window for this surface.

`-lh` = Set the most recently defined surface's colormap legend window to have a horizontal instead of vertical layout.

`-lm landmark_filename` = read from the file `landmark_filename` a set of coronary arteries, or any other landmark information stored as a series of points, with a radius associated with each. See section 6.5 below for details.

`-ll leadlinks-filename` = file in *leadlinks* format containing a list of the node locations that correspond to a subset of the leads, *e.g.,* the lead locations on the torso surface that correspond to the standard ECG leads. The point of identifying such leads is to display them with their own markings, either as spheres or with the lead number (typically not the same as the node number). For more information, see the menu options in Section 8.2.3 that determine the form of the display markings and Section 6.4 for more information on leadlinks files.

4.4 Scalar Data parameters

To display scalar data values on the geometry, we must specify the source of the data and how to link them to the geometry. As with the geometry, all arguments specified between two occurrences of `-f` in the command line refer to the currently valid surface. Within pairs of `-f` options, there can be only a single instance of any of the following options:

`-p potfilename` = filename for the potential and current data files. The legal file types for scalar data are:

1. Pot files (.pot) (see Section 6.3.1 are text-based files which contain one file per time instance.
2. matlab files (.mat) are also stored in binary format. Matlab files can also contain multiple time series.

The `-s` option determines how many frames to load. In the case of pot files, this controls which pot files to open (If `-s` is omitted, it will only open the pot file specified). For binary files, the `-s` option specifies the start and end frame numbers to be read from the file. With no `-s` option, *map3d* will read in all time instants from the file. Note also that if you omit the extension, as with geometry files, it will try to match a .pot or .mat extension for you.

For files with multiple time series (matlab or tsdf containers), you may specify the time series by the command line with the "@" suffix appended to the filename followed by the time series index within the file.

eg., `-p file.mat@1` reads the first time series and *eg.,* `-p file.tsdfc@2` reads the second time series.

`-s num1 num2` = range of frame numbers to read. If we are reading data from .pot or .grad files, *map3d* appends each of the numbers between `num1` and `num2` to the value of `potfilename` to make complete pot filenames. However, you must run *map3d* with the full pot filename (one of the pot files in the series).
eg., `-p good-map001.pot -s 1 3` expands to:
`good-map001.pot good-map002.pot good-map003.pot`
If we are reading from a matlab (.mat) data file, *map3d* will read frames `num1` to `num2` from the file.

`-i increment` = difference between each frame number. With the last few versions, this would still read in all the frames, but this version acts more like the versions prior to that, and subsamples the data.

`-ph maxpotval` = maximum data value in "user" scaling mode. This sets one option for setting the range used in scaling the data value to colours and contours. You can select other ranges from the menu and can select this one again with Scaling-¿Range-¿Command-line specified range.

`-pl minpotval` = minimum data value in "user" scaling mode.

`-cs contour-spacing` = spacing between contours set by the user. This provides a menu option for selecting contours by setting a constant spacing rather than deriving the spacing from the desired number of contours and the range of data values. Note that the spacing will not always be a the command-line set value - *map3d* will divide the range by the specified value and set the number of contours as that number, and then determine the contour values by using that number of contours with the currently- selected scaling function. You can select other numbers of contours from the menu and can select this again with Contours-¿Number of Contours-¿Command-line spacing

`-ps scaleval` = scaling value by which *map3d* multiplies each potential value as it reads from the file(s). This option tries to make use of any unit information available in a time series data file and alters the unit value available to *map3d* for display. The resulting scaling of the data is permanent for the current instance of *map3d*.

`-ch channels-filename` = file in *channels* format containing an entry for each node in the geometry which points to the associated location in the data array. The value of this pointer is also the number that is written next to node locations when channel numbers are displayed. See section 6.4 for more information on the channels file format.

`-lm landmarks-filename` = file in *landmarks* format containing a set of landmark segments, divided into categories. Each category has a word depicting the landmark type. Each lines within the categories contains three points (x,y,z) and an associated radius, which may have a different effect based on the type of landmark. See section 6.1.4.

`-ff fidfile` = Ascii file containing fiducial information. Information may be specified for each node for an arbitrary set of fiducial data. See section 6.3.3.

`-sl surfnum` = surface number to which the scaling for this surface is to be slaved. The idea here is to have surfaces locked in the way they scale and display the data; in this way, one can compare colors across surfaces to determine relative values of the local scalar data.

`-t timesignal-lead-number` = number of the node to be used for the display of a time signal in its own window. The number refers to either a node number in the geometry or, if a leadlinks file is present, the *lead* number. This command is optionally used in conjunction with the -at command, to specify a node and place its window accordingly. If the -at option is not present, map3d will choose a default window location. Multiple invocations of this option are possible for each surface, providing the option to open several windows per surface. At any time during the operation of the *map3d* the user can select a new node via the pick mode menu item and have the time signal from that node displayed (see Section 8.6 for details).

`-at xmin xmax ymin ymax` = set the absolute location in pixels of a time signal window associated with the current surface. As with the `-as` option, the origin is in the lower left corner of the screen and the full screen resolution of an SGI screen with 19-inch monitor typically supports 1280 by 1024 pixels. This command is optionally used in conjunction with the -t command, to specify a node and place its window accordingly. If the -t option is not present, map3d will choose a default node (the first node in the geometry). Multiple invocations of this option are possible for each surface, providing the option to open several windows per surface.

4.5 Other usage parameters

map3d accepts many other command-line parameters to customize the display. These are optimally used in a script, but the following sections showcase some of the options not described above. Some of these options will only be pertinent when used from a script file (see Section 5).

4.5.1 Additional global parameters

These parameters affect all surfaces or windows in *map3d*.

`-bw borderWidth titleHeight` Specifies the border width and the height of the title bar in pixels used in window placement calculations.

-l general transformation frame Specifies whether the general, transformation, and frame locks should be turned on. Specify a 1 or a 0 for each, where 1 signifies to enable the lock, and 0 to disable.

-pm mode Specifies the pick mode, where mode is an integer.

```
0  New Window Mode
1  Refresh Window Mode
2  Node Info Mode
3  Triangle Info Mode
4  Triangulate Mode
5  Flip Triangle Mode
6  Edit Node Mode
7  Edit Landmark Mode
8  Delete Node Mode
```

-sc range function mapping Describes the scaling model, with the specified scaling range, function, and mapping, each being an integer.

```
Range (data min and max is relative to):
0  Local
1  All frames in current surface
2  All surfaces in current frame
3  All surfaces and frames
4  All surfaces in current group in current frame
5  All surfaces and frames in current group
6  Slave scaling in current frame
7  Scave scaling in all frames

Function
0  Linear
1  Exponential
2  Logarithmic
3  Lab7
4  Lab13

Mapping
0  Symmetric
1  Separate
2  True
3  About Midpoint
```

4.5.2 Additional surface parameters

These options affect the current surface, if placed after the first [-f] on the command line, or all surfaces if placed before.

-sf size Size of the small font, from 0-9

-mf size Size of the medium font, from 0-9

-lf size Size of the large font, from 0-9

-cm map Which colormap the data will use, where map is an integer:

```
0  Rainbow
1  Green to Red
2  Grayscale
3  Jet
```

-sm mode Selects the shading mode of surface data, where mode is an integer:

```
0  No shading
1  Flat
2  Gouraud
3  Banded
4  Textured Gouraud
```

-rm mode Selects the mesh rendering mode, mode being an integer:

```
0  No rendering (mesh is not displayed)
1  Points Only
2  Mesh Elements
3  Mesh Connectivity
4  Elements and Connectivity
5  Points and Connectivity
6  Render only the mesh elements with no surface data
```

-ic num Whether or not to invert the colormap, 1 meaning yes, 0 meaning no

-nco num Sets the number of contours to num

-rq w x y z Sets the rotation of the surface to the quaternion formed by (w,x,y,z)

-tc x y z Sets the translation of the surface to (x,y,z)

-zf vfov Sets the "zoom factor" or field of view to vfov

-el num Whether or not to light the surface, 1 meaning yes, 0 meaning no

-ef num Whether or not to display the surface with fog, 1 meaning yes, 0 meaning no

-gn num Assigns the group number to num for group scaling

-sco num Whether or not to show contours, 1 meaning yes, 0 meaning no

-nc num Whether or not to show the negative contours as dashes, 1 meaning yes, 0 meaning no

-x num Whether or not to show axes, 1 meaning yes, 0 meaning no

-xc r g b Specifies the axes' color, with RGB numbers from 0-255

-sit num Whether or not to show informational text, 1 meaning yes, 0 meaning no

-sli num Whether or not to show the lock icons, 1 meaning yes, 0 meaning no

-nma sphere mark value Settings for Node Marking: all. Sphere indicates whether to draw spheres for all nodes, 1 meaning yes, 0 meaning no. Value indicates whether to color the spheres based on their data value, 1 meaning yes, 0 meaning no. Mark indicates what to display at the nodes, mark being an integer:

```
0  nothing
1  Node number
2  Channel number
3  Data value
```

-nme sphere mark Settings for Node Marking: extrema, where sphere and mark have the same meanings as [-nma], except the affect the extrema nodes

-nmp sphere mark Settings for Node Marking: pick, where sphere and mark have the same meanings as [-nma], except the affect only the nodes that have an active "Time Series" Window open.

-nml sphere mark Settings for Node Marking: leadlinks, where sphere and mark have the same meanings as [-nma], except the affect only the nodes that are represented in the active leadlinks file, except that a mark of 3 will display the lead label instead of the data value.

-Sll value Sets the size of the leadlinks points, from 0-9.

-Cll r g b Sets the color of the leadlinks points, with RGB numbers from 0-255

-rf num Whether or not to draw fiducials, 1 meaning yes, 0 meaning no.

5 Script Files

Using script files to control map3d has numerous advantages, for example:

1. complex layouts and specifications can be created and then held for later reuse,
2. execution of the program reduces to a single word that starts the script,
3. scripts are shell programs and can include logic and computation steps that automate the execution of the program; the user can even interact with the script and control one script to execute many different actions.

5.1 What are script files?

A script files are simple programs written in the language of the Unix shell. There are actually several languages, one for each type of shell, and the user is free to select. At the CVRTI we have decided to use the Bourne shell for script programming (and the Korn shell for interactive use) and so all scripts will assume the associate language conventions. The shell script language is much simpler to use and learn than a complete, general purpose language such as C or Fortran, but is very well suited to executing Unix commands; in fact, the script files consist mostly of lists of commands as you might enter them at the Unix prompt. Even more simply, a script file can consist of nothing more than the list of commands you would need to type to execute the same task from the system prompt.

5.2 Automatically generating script files

The easiest way to make a script file is to have *map3d* do it for you. Open a set of surface, and arrange things how like them, and select "Script file" or "Windows batch file" Save menu. In the script section of the dialog, select a filename and the type of script file, and push "save". This will generate a script file that, when run, will open *map3d* very closely (if not identical) to how you had it before.

5.3 How to make script files manually

Script files are simple text files and so are usually created with an editor such as emacs. You can, however, also generate a script file from a program, or even another script. But all script files can be read and edited by emacs and this is the way most are composed.

To learn about the full range of possibilities in script files requires some study of a book such as "Unix Shell Programming" by Kochan and Wood but the skills needed to make *map3d* script files are much more modest; any book on Unix will contain enough information for this. The instructions and examples below may be enough for many users.

Here are some rules and tips that apply to script files:

Use a new line for each command This is not a requirement but makes for simpler files that are easier to read and edit. If the command is longer than one line, then use a continuation character "
", *e.g.,*backslash

```
map3d -f geomfilename.fac \
       -p potfilename.tsdf \
      -cl channelsfilename
```

Make sure that there are no characters (even blank spaces) after the continuation character!!! This has to be the most frequent error when the script file fails to run or stops abruptly.

Make script files executable Script files can be executed by typing `. scriptfile` but the simplest thing is to make then executable files so that they work simply by typing their names. To do this, use the `chmod` command as follows:

```
chmod 755 script_filename
```

Use the .sh extension for scripts This convention makes it easy to recognize shell scripts as such, but also invokes some editor help when you edit the file in emacs. The mode will switch to shell (much like Fortran or C mode when editing programs with .f and .c extensions) and has some automatic tabbing and layout tools that can be helpful.

Variables in scripts The shell script is a language and like any computer language there are variables. To define a variable, simply use it and equate it to a value, *e.g.,*

```
varname=2
varname="some text"
varname=a_name
```

Do not leave any spaces around the "=" sign or the command will fail and set the variable to an empty string.

Once defined, the variables can be used elsewhere in the script as follows:

```
geomdir=/u/macleod/torso/geom
geomfile=datorso.fac
datafile=dipole.tsdf
map3d -f ${geomdir}/${geomfile} -p $datafile
```

The curly braces are required when the variable name is concatenated with other text of variable names but is optional otherwise. To concatenate text and variables you simply write them together (*e.g., geomdir*/`geomfile.pts` concatenates the two variables with a "/" and the extension ".pts".

Environment variables in scripts All the environment variables are available and can be set in the script. For example:

```
mydir=${HOME}
```

sets the variable `$mydir` equal to the user's home directory. Likewise,

```
MAP3D_DATAPATH=/scratch/bt2feb93/
export MAP3D_DATAPATH
```

defines and "exports" the environment variable used by map3d to find .pak files.

A note on scripting on Windows Windows scripting is a little different.

Windows environment variables can be set as follows:

```
set VAR=VALUE
```

and referred to as %VAR% later.

The *map3d* command must all be on one line - it will not accept \as a continuation character

6 Input files

In this section, we describe the contents and formats of all the input files that *map3d* uses to get geometry, data, and much more.

6.1 Geometry input files

The input of geometric data for *map3d* occurs via files and we support three different formats at present. The simplest (and oldest) is a set of ASCII files that contain the points or nodes of the geometry—stored in a file with the extension .pts—and the connectivities that described polygonal links between nodes—stored as line segments (.seg files), triangles (.fac files), and tetrahedra (.tetra files). To satisfy a need for more comprehensive and compact storage of geometry information, we have developed a binary file format and created the *graphicsio* library to manage these files. Finally, in recognition of the ubiquity of MATLAB, as of version 6.1, there is support for reading .mat files, which have an internal structure that included node and connectivity information. Below, we describe each of these files and how *map3d* uses them.

6.1.1 Points (.pts) file

The characteristics of a .pts file are as follows:

1. ASCII file, no special characters permitted;
2. Each line contains one triplet, ordered as x, y, and z values; one or more spaces between values, which are assumed to be real, floating point numbers;
3. Each line may also optionally contain a group number as a fourth element (although at present, *map3d* does not use this group information);
4. the order of points in the file is the implicit order of the nodes in the geometry; connectivities are based on this ordering.

6.1.2 Triangle (.fac) files

The characteristics of a .fac file are as follows:

1. ASCII file, no special characters permitted;
2. Each line contains a triplet of integer values pointing to the nodes of the geometry. **Node numbers begin at 1 not 0!**;
3. The order of triangle vertices (nodes) is not strictly controlled, however, it is recommended that order reflect a common convention in graphics—a counterclockwise sequence of vertices when viewed from the **outside** of the triangle;
4. Each line may contain an optional fourth values which is the group number for the triangle (not used by *map3d*);
5. Order of triangles in the file is not meaningful except for internal bookkeeping; user will notice ordering only when a triangle is picked for interrogation.

6.1.3 MATLAB geometry file support

map3d can read .mat files generated by MATLAB as long as they are organized according to the following guidelines:

1. Each separate surface is either a structure (See the MATLAB documentation for structures). To include multiple surfaces requires an array of structures.
2. Within a surface structure, the following fields contain the geometry:

- *.pts or .node* contains the node locations, usually in a $3 \times N$ array (although *map3d* will check and accept either $3 \times N$ or its transpose, $N \times 3$), where N is the number of nodes.
- *.pts_mv* contains animated node locations, usually in a $multiple - of - 3 \times N$ array (although *map3d* will check the transpose like with pts). pts_mv and pts are mutually exclusive. See Section 7.3.1 for the description. Each set of 3 columns will be the node locations for a particular index of time.
- *.fac or .face* contains the triangle connectivities, usually in a $3 \times M$ array (again, *map3d* will accept the transpose) where M is the number of triangles.
- *.seg or .edge* contains the line segment connectivities,
- *.tet, .tetra, or .cell* contains tetrahedral connectivities, and
- *.channels* contains the channel information in a one-dimensional vector, in which each element of the vector points to the associated data channel.

To prepare a structured .mat file is very simple, for example using the following commands:

```
>> geom.pts = ptsarray;
>> geom.fac = facarray;
>> geom.channels = 100:164
>> save mygeom.mat geom
```

where `ptsarray` is a $3 \times N$ array defined to contain the node locations, `facarray` is a $3 \times M$ array of triangle connectivities, and `mygeom.mat` is the name of the resulting .mat file. The channels information indicates that there are 64 nodes in the geometry and they expect to get time signals from channels 100–164 of a data file. (See Section 6.4 for more information on channels.

6.1.4 Landmark geometry file support

map3d can also read geometry from a landmark file (See Section 6.5 below), where you specify a series of connected points and radii. *map3d* will automatically connect and triangulate them, and will also associate scalar data with them. Note that currently there is no channels support for landmark geometry.

6.2 Command line control of geometry files

In *map3d* the -f option determines in which files the geometry is to be found. Starting from the filename that follows -f, which may or may not include a file extension, the program looks for all possible candidate geometry files and queries the user for resolution of any ambiguities. Thus, with the arguments `map3d -f myfile`, *map3d*3d looks for `myfile.mat`, `myfile.pts`, `myfile.fac`, *etc,* and tries to resolve things so that a valid geometry description is found. It is possible to direct this process by typing the geometry filename with an extension according to the following rules:

Extension	Effect
none	look for files with the extensions .pts, .fac, .tetra, and .mat, and if an incompatible set are present (*e.g.,* both .pts and .mat), ask user which to take
.pts	take only the .pts file and ignore any connectivity or .mat file
.fac	take .pts and .fac and ignore any .mat files present.
.mat	take the MATLAB file and ignore any others present.

Note that a MATLAB file could contain both geometry and potentials; it just has to be referenced as a geometry file and a potentials file.

6.3 Scalar data files

There are two ways of storing scalar values (typically electric potential in our applications) so that *map3d* can recognize and read them. One is a simple ASCII file, and the other is MATLAB.

6.3.1 .pot files

One way to package the scalar data values that are assigned to the points in the geometry is the .pot file. In the default condition, the scalar values in the .pot files are ordered in the same way as the node points in the geometry file with simple one-to-one assignment. With a channels file, it is possible to remap this assignment, as described in Section 6.4).

The rules for .pot files are:

1. Each line of the files contains one scalar data value, assumed to be a real number in single precision floating point format.
2. The order of the values within the file must either agree with that of the associated set of nodes or a channels file must be supplied to redirect the links between potential value and nodes.
3. Each .pot file *must* end with a blank line.
4. A single .pot file can contain only the data values associated with a single surface at a single instant in time. To represent a sequence of time steps (frames) requires a sequence of files, typically with filenames ending in a three-digit series, *e.g.,* `mapdata001.pot`, `mapdata002.pot`, `mapdata003.pot`, ... Section 4.4 explains how to specify such a series of .pot files to *map3d* and Section 4.1 shows examples.
5. The extension .pot *must* be used.

6.3.2 MATLAB data file support

You can also store and read scalar values in .mat files, as a structure with a single field called ".potvals", that contains a $N \times M$ array, where N is the number of data channels and M is the number of time instants. There are additional fields in the structure —the complete list is as follows:

- *.potvals, .data, .field, or .scalarfield* scalar values as $N \times M$ array, where N is the number of data channels and M is the number of time instants.
- *.unit* the type of units for the data, "um" for microvolts, "mv" for millivolts and "V" for volts.
- *.label* the name of the time series. This is optional, but is useful in identifying the time series, particularly from a multi-time-series file.
- *.fids* a structure (or array of structures) containing fiducial time markers for the time series (see below for details of fidicual storage).

Note that only the 'potvals' field is required. A matlab array may be one instance of these fields, a cell or struct array of them, or simply an $N \times M$ array of data. To read a matlab file with many time time series, specify the time series on the command line with the '@' symbol (See Section 4.4) or specify the time series in the file browser. It will be shown by the label specified in the matlab file.

The commands to make a suitable .mat file are very easy in MATLAB, for example, to load 128 channels of time signals with unit of millivolt from an array `sockinfo` in MATLAB to a file called `mysockdata.mat`

```
>> sockdata.potvals   = sockinfo(1:128,:);
>> sockdata.unit = 'mV'
>> sockdata.lavel = 'A set of cool sock data'
>> save mysockdata.mat sockdata
```

6.3.3 Fiducial (ascii) files

A fiducial can be input currently also in two ways: via a .fids file, a simple ASCII file containing the values for each node, or by means of the MATLAB file that contains the time series data.

MATLAB file format for fiducials The fiducial information in a MATLAB time series data file is stored in a array of structures called fids, where each element of the array represents one set of fiducials. This way, for example, there can be a set of N activation times, another set of N recovery times, and a set of N Q-onset times. Each element of the array is, in itself, a structure, so that the set of fiducials contains enough information for *map3d* to do a useful display.

The legal elements of a fids structure are as follows:

- *.type* the type of fiducial (see table below).
- *.value* the array of fiducial values, one for each channel of the data file.

Here is an example of creating the structures for some fiducials in MATLAB:

```
matdata.potvals = potvals;
matdata.unit = unit;
matdata.label = label;
matdata.fids(1).type = 10
matdata.fids(1).value = fidvals(:,1);
matdata.fids(2).type = 13;
matdata.fids(2).value = fidvals(:,2);
save (outfilename, 'matdata');
```

The first three lines set up the time series data, as we have seen above. The lines 4 and 5 create a set of fiducials of type 10 and loads the values into that part of the data structure. Lines 6 and 7 do the same thing for a second set of fiducials, which have type 13. The final line saves the contents of the structure into a file.

The type value for fiducials is important as *map3d* associates colors to fiducial types to provide consistent display. Moreover, each type has an associated text label that *map3d* uses in the interface. The list of current fiducial types and their numbers is as follows:

Fiducial Types	
Number	Type
0	pon, pstart
1	poff, pend
2	qon, qrsstart, qrson
3	rpeak, qrspeak
4	soff, qrsend, qrsoff
5	stoff, tstart, ton
6	tpeak
7	toff
8	actplus
9	actminus
10	act
11	recplus
12	recminus
13	rec
14	ref, reference
15	jpt, jpoint
16	baseline
30	pacing

ASCII file format for fiducials The characteristics of a .fids file are as follows:

1. ASCII file

2. must have the following at the top of the file, on each line:
   (a) number of time series, e.g., 1 (*map3d* only allows 1)
   (b) series number (space) pak number
   (c) number of nodes (space) list of fiducial types
3. each successive line contains the node number followed by a list of fiducial values, one corresponding to each type specified on the line with the node numbers

```
Example:
1
1 -1
256 activation recovery
1 8 212
2 16 225
3 9 214
...
255 39 248
256 25 237
```

6.4 Channels and leadlinks

6.4.1 Description of leadlinks and channels information

Channels and leadlinks files, and the arrays they contain, are identical in structure, but they have important **functional differences**. A run of *map3d* may require both, either, or neither of these, depending on the structure of the data files and geometries. The basic function of both channels and leadlinks information is to offer linkages between nodes in the geometry and the data that is to be associated with those nodes. The first file type, the channels file, links the nodes in the geometry to specific time signals in a data file; without channel mapping, the only possible assumption is that each node i in the geometry corresponds to the same time signal i in the data file. Any other linkage of geometry and data channel requires there to be channel information, typically either from a separate .channels file or stored with the binary .mat file as an associated scalar value for each node.

Leadlinks are purely for visualization and describe the connection between "leads", a measurement concept, with "nodes", a geometric location in space. In electrocardiography, for example, a lead is the algebraic difference between two measured potentials, one of which is the reference; "unipolar" leads have a reference composed of the sum of the limb electrode potentials. It is often useful to mark the locations of these leads on the geometry, which often contains many more nodes than there are leads. The most frequent use to date has been to mark the locations of the standard precordial ECG leads within the context of high resolution body surface mapping that uses from 32–192 electrodes. Another common application is to mark subsets of a geometry that correspond to measurement sites (values at the remaining nodes are typically the product of interpolation). In summary, leadlinks allow *map3d* to mark specific nodes that may have special meaning to the observer.

Figure 1 shows an example of lead and channels information and their effect on *map3d*. See the figure caption for details.

map3d handles this information in the following manner:

channels The *channels* information links nodes in the geometry to individual channels or time series in the data file. For example, the data values associated with node k in the geometry are located in the data location specified by the channel array value channels(k). If channels(k) < 0, then there is no valid data for node k.

Note that ***map3d* uses the channels arrays when loading scalar data into the internal storage arrays!** Hence, the action of the channel mapping is not reversible. Should geometry nodes and data channels match one-to-one, there is no need for a channels array. It is also possible to define via a channels array the situation in which a single data channel belongs to two (or more)

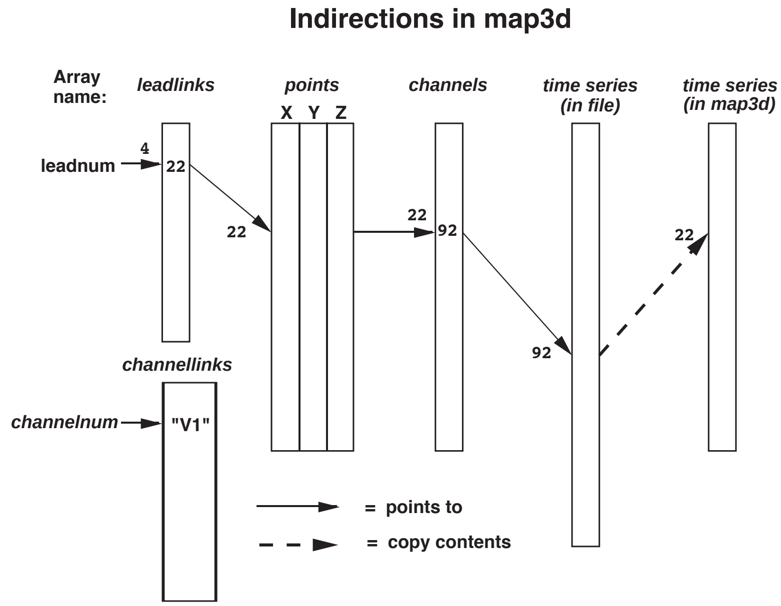

Figure 1: Example of the indirection possible in *map3d* through the use of *leadlinks channels*, and *channellinks*. Lead number 4 points, via the *leadlinks* array to node number 22. This, in turn, points via the *channels* array to location 92 in the multiplexed data buffer, which causes the values at time signal 92 to be loaded into location 22 in the internal data array (and displayed by *map3d*). In a separate, *channellinks* array, shown below the leadlinks array, the entry in lead 4 says that that lead should actually be labeled "V_1".

nodes in the geometry. The most frequent example of this occurs when three-dimensional geometries are "unwrapped" into surfaces in which what was a single edge is split and thus present at both ends of the surface.

leadlinks The *leadlinks* information is primarily used to identify and mark measurement lead specific within the geometry. The typical use is to select a subset of the nodes to identify the measurement sites—values at other sites might be interpolated or otherwise computed. Leadlinks also provide a means to renumber the labels on the nodes of the geometry in order to, for example, reproduce the numbering scheme used in an experimental setup.

In the leadlinks array each entry refers, by its location in the array, to a particular lead #; the value at that location in the array gives the number of the node in the geometry file to be associated with this lead. For example if lead 4 has a leadlinks entry of 22 (leadlinks(4) = 22), then *map3d* will display node 22 in the geometry as "4", not "22" whenever node marking with *leads* is selected).

channellinks There is an extension of the basic scheme which includes a further level of redirection for giving the leads explicit text labels. Channel links are stored as a array of strings, one for each node of the geometry. The channellinks file is organized similarly to a leadlinks file, with each line containing information for one node. However, each line consists of two values, 1) the number of the associated channel and 2) the text string to be used as the label when *map3d* marks the nodes with channel numbers.

Hence we have the situation of a node number K in the geometry displaying time signals from channel L in the scalar data, but labeled with string "XXX".

6.4.2 Source of leadlink and channel information

The sources of channels, leadlinks, and channellinks information are files, or parts of files, as outlined in the following paragraphs.

In .mat geometry files MATLAB geometry files can also contain an *channels* array is stored as a .channels field in the structure. See Section 6.1.3 for more details.

In .leadlinks files A .leadlinks file is an ASCII file, the first record of which contains a line `nnn leads`, where `nnn` is the number of leads to be described in the file (and also one less than the total number of lines in the file). Each following record contains two integer values:

1. the first number is the number of the lead, as it should appear in any labeling of the associated node.
2. the second entry in each row is the value of the associated node number in the geometry.

For example, the file for a surface which reads:

```
32 Leads
1   1
2   42
4   31
7   65
.   .
.   .
.   .
32  11     <---- 32nd entry in the file, at line 33 of the file.
```

indicates that there are 32 leads to be linked (the geometry can, often does, contain more than 32 nodes), and that lead #1 is called lead "1" and is node 1 in the geometry file. Lead #2 is at node 42, lead #3 is called "4" and is found at node 31. Likewise, lead #4 is called "7", and is located at node 65, and so on, up to lead #32, called "32", at node 11.

Nodes listed in a *leadlinks* file that is passed to *map3d* with the `-ll` option can be marked in a number of ways, described more fully in Table 11 in Section 8.2.3.

In .channels files A *channels* file is an ASCII file, the first record of which contain a line `nnn nodes`, where `nnn` is the number of nodes to be described in the file (and also one less than the total number of lines in the file). There is one line in the file for each node of the geometry to which we wish to associate scalar data. Each following record contains two integer values:

1. the first number is simply a running counter that indicates the node number with which to associated the second value in the row.
2. The second value in each row is the *channel* number for that node; a negative number signifies a node to which there is no data associated.

For example, the file for a surface which reads:

```
352 Nodes
1    123
2    632
.    .
.    .
22   -1
23   432
.    .
.    .
352  12
```

indicates that there are 352 leads to be linked, and that the data value for the first node is located at location 123 of the data file. For node 2, the data value is to be found at location 632, and so on. Node 22 does not have any scalar data associated with it.

6.4.3 Display of lead/channel information

To see how *map3d* can display the node and lead information, see Sections 7.8 and 8.

6.5 Landmark files

Landmark files contain information for visual cues or landmarks that *map3d* can draw over the surfaces in order to aid and orient the viewer. Initial use was primarily for coronary arteries, but the idea has expanded to incorporate a number of different orientation landmarks. The original coronary artery class of landmarks requires only that each can be described as a series of connected points, with a radius defined for each point. The coronary landmark is then displayed as a faceted "pipe" linearly connecting the points at the centre, with a radius, also linearly interpolated between points, determining the size of the pipe. The landmark file can contain numerous, individual segments of such pipe-work, each of which is drawn separately.

Other classes of landmarks are described below, but all of them can be described in a file with the following general format:

Line number	Contents	Comments
1	nsegs	number of landmark segments in the file
2	1 type nsegpts [segindex] [segcolor]	segment number (1), type, number of points, with optional index number and color (color is a RGB triple, 0-255 each)
3	X, Y, Z, radius [label]	point location and radius of point 1, with optional label to be drawn at the point
4	X, Y, Z, radius [label]	point location and radius of point 2
.	.	.
.	.	.
nsegpts + 2	X, Y, Z, radius	point location and radius of last point in segment 1
.	2 type nsegpts [segindex] [segcolor]	segment number (2), type, number of points, with optional index number and color
.	X, Y, Z, radius [label]	point location and radius of point 1
.	X, Y, Z, radius [label]	point location and radius of point 2
.	.	.
.	.	.
.	X, Y, Z, radius [label]	point location and radius of last point in segment 2
.	.	.
.	.	.
.	.	.
.	.	.

The landmark types defined to date are the following:

Name	Graph. object	Description
Artery	faceted pipe	a coronary artery/vein segment
Occlusion	coloured sphere	an experimental occlusion that could be open and closed
Closure	coloured sphere	a permanent occlusion that cannot be opened
Stimulus	coloured sphere	a stimulus site
Lead	coloured sphere	a particular electrode or lead location
Plane	rectangular parallolop-iped	a visible (but not functional) cutting plane through the geometry (Note: do not confuse this with the cutting planes that *map3d* provides for slicing through the geometry).
Rod	lines	rod inserted into needle track to digitize needle electrode locations.
PaceNeedle	sphere	location of a pacing needle entry point
Cath	faceted pipe	location of catheter in a vessel
Fiber	arrow	local fiber direction indicator
RecNeedle	sphere	location of recording needle entry point
Cannula	tube	coronary vessel cannulus

Specifying snare, closure, and stimulus requires a single point in the landmarks file, and the resulting sphere is coloured according to a set of values defined in the drawsurflmarks.c routine. At present, the values used are:

Occlusion	cyan
Closure	blue
Stimulus	yellow

and they are adjustable by the user, either via the Landmark menu, or by specifying a color for the segment

To specify a plane landmark requires three "points"

Point	X,Y,Z	Radius
1	First point of plane	Radius of the plane
2	Second point of plane	Thickness of the plane
3	Third point of plane	not used

The plane is drawn as a polygon with the number of sides controlled by a program variable.

Filename conventions: The standard extension of a landmark file is `.lmarks` and the filename is specified by the `-lm` parameter for each surface.

Control of landmark display: for details of how to control the display of landmarks, see Section 8.7.

7 Display features

This section describes the displays that *map3d* generates and what they mean; for specific information on how to control *map3d* and the displays, see Section 8.

7.1 Multiple surfaces

The idea of *map3d* has always been to display multiple sets of data on multiple surfaces; the limitation has been how much flexibility to include in a single invocation of the program. This version of *map3d*, as opposed to previous versions, can now handle multiple windows each with multiple surfaces. Surfaces can be moved between windows (see Section 8.3.1) When *map3d* displays multiple surfaces, each can exist in its own full window with its own border and window title bar, or, *map3d* can build a single main window with multiple sub-windows inside the main window. The user can reposition and resize each of these sub-windows using the Alt(Meta) key and the left and middle mouse buttons respectively. To create this layout of main window and frameless sub-windows, use the `-b` (borderless windows) option when launching *map3d* as described in Section 4.3.

7.2 Surface display

The basic forms of display of the surfaces are

- nodes or points from each surface
- connectivity mesh
- shaded surfaces based on either the geometry or the associated scalar values, with a number of different rendering options.
- landmarks superimposed on the surface display

7.3 Mesh Rendering

Often the purpose of *map3d* is to render a geometry consisting of nodes and connectivities and there are several basic modes of rendering this information.

Points: display just the nodes of the geometry as dots or marked with spheres.

Connectivities: display the connectivity information for the mesh as lines joining the nodes.

Elements: treat each polygon in the mesh as an element and render it in a way that shows its surface; for triangles, simple render each triangle surface; for tetrahedra there is no specific rendering in this version of *map3d*.

Elements and connectivities: *map3d* also supports a hybrid mode of rendering that shows outward facing triangles (using the convention of counterclockwise ordering) as elements but backwards facing triangles as connectivities.

map3d also has the ability to render all elements with a lighting model. This is especially useful for displaying the elements of the mesh. Additional controls to note are depth cueing, which can reveal the depth relationships between elements of the mesh.

map3d will render the mesh as best as it can to match the surface data settings. I.e., if the surface data is set to render flat, the mesh will render flat, and if the surface data is set to render gouraud or band shaded, then the mesh will render shaded.

map3d will render the mesh and surface data best if all the triangles are front-facing. If some of the triangles are not front-facing, they will look out of place when gouraud shading is on. However, this is not enough to determine which triangle is back-facing. Since *map3d* renders both front- and back-facing triangles, the only distinguishing factor is the normal vector when being drawn in gouraud shading. If all the triangles are back-facing, none of them will look out of place. If you're trying to get all triangles to face the same direction, then it's best to render the mesh as Elements and Connectivities, as back-facing triangles will explicitly be drawn as connectivities.

7.3.1 Animated Mesh Rendering

Sometimes a geometry's nodes change locations over time. *map3d* supports rendering animated geometry. See Section 6.1.3 for the file specification. The animated nodes work hand in hand with the surface data frames, and are divided proportionally, thus if there are 1000 data frames and 100 animated geometry node sets, each node set will display for 10 frames.

7.4 Surface Data Display

The main use of *map3d* is to display scalar data associated with geometry and there are numerous options and controls to facilitate this. The two basic ways of conveying scalar value information are as shaded surfaces and contour lines and *map3d* supports each separately, as well as in combination. For surface shading, there are several basic rendering modes:

Flat: each triangle received a single color that depends on the mean value of the scalar value over that triangle.

Gouraud: the colour of each triangle values linearly with the value at each of the vertices. The current version uses texture mapping to achieve more desirable results, but if your hardware does not support texture mapping, you can toggle it with the g-key.

Banded: the regions between contour lines have a constant color, even if the contour lines are not visible.

Contours: this can be a separate rendering mode, or combined with any of the three modes above. Contours are lines that trace iso-values over the surface of the geometry.

7.4.1 Data scaling

There is a wide variety of options available for mapping scalar values to colour and contour levels. One can picture the process as based on four facets:

Extrema: the extrema of the data and the selected colour maps determine the basic parameters of how value maps to color. *map3d* maintains a detailed list of data extrema organized both by time signal, time instant and by surface. Thus it is possible to determine extrema based on just the most local of conditions—a particular frame and surface—or by more global conditions—the full range of frames or the full set of surfaces.

Scaling function: the mapping between value and color occurs according to some mathematical function, the simplest of which is linear. The scaling function uses the selected extrema and describes a complete mapping between value and color.

Mapping: by scale mapping, we mean how the translation from value to color treats positive and negative values. We may choose to map uniformly between the extrema or to apply different extrema or functions to the positive and negative values.

Color maps: the color displayed for a particular scalar value depends on the actual range of colors and their order in the color map.

map3d can adjust all four facets of the scaling to create a wide range of displays. Note that all surfaces conform to the first three scaling options (there is only one scaling range, function, and mapping that corresponds to all surfaces). We also chose to limit some of these options, however, in an effort to create reproducible displays that reflect standard within the field. Of course, we chose our field, electrocardiography, as the basis, a fact for which we make no apologies and simply encourage others to make similar choices for their own field and implement *map3d* accordingly. Subsequent versions of *map3d* will support this flexibility.

Below are the specific choices that *map3d* offers to control data scaling and display

Scale range *map3d* supports several selections of range over which to look for extrema. In *local* range, only the data presently visible are scanned for extrema—this is the default. In the full *global* range, all the data in the entire dataset are used, even those not presently visible on the display. In between these cases, one can have global in time and local in space, *i.e.,* we scale each surface separately but use all time values for that surface. Or one can select local in time and global in space, in which *map3d* scans all surfaces for the data extrema, but for each time instant separately.

The user can also select group scaling, where he assigns surfaces to groups and the range is based on the group min/max (either local in time or global). Groups are assigned by the menu. The user can also do slave scaling, where he assigns one surface (slave) to another's (master) range. The slave surfaces are currently only set through the command line, by placing a -sl num (where num is the surface number of the master) after declaring the slave surface. See Section 8.2.3 and Section 4.4 for details.

If these options are not suitable, the user can select his own scaling scope for maximum and minimum values. This can be set via the command line (see `-pl` and `-ph` input parameters in Section 4). or in the Contour Spacing dialog (see Section 8.5.5). While the rest of the scaling range options are set once for all surfaces, whether or not an individual surface corresponds to the default range can be selected through the Contour Spacing dialog as well.

Scale function The scale model describes the way in which scalar data are mapped to colours (or contours). The present choice is linear, but the next version of *map3d* will include: **linear** model, which simply maps the data to a range of colours in a completely linear fashion, *i.e.,* $colour = K\phi$; the **logarithmic** model, which highlights the lower level data values at the cost of poorer resolution at the higher levels *i.e.,* $colour = A\log(\phi) + B$; and the **exponential** model, which does the opposite, compressing the smaller levels and expanding the higher ones to span a wider colour range, *i.e.,* $colour = Ae^{B\phi}$.

The two schemes with fixed numbers of contours, *log/7-levels* and *log/13-levels* both map the upper decade (ϕ_{max} to $\phi_{max}/10.$) of the potential data range into a fixed set of logarithmically spaced values. These values are composed of a mantissa from the standard E6 (1.0, 1.5, 2.2, 3.3, 4.7, 6.8, and 10.) and E12 (1.0, 1.2, 1.5, 1.8, 2.2, 2.7, 3.3, 3.9, 4.7, 5.6, 6.8, 8.2, and 10.) number series, and an exponent such that the largest mantissa falls into the range 1.0 to 10. Hence as long as the extrema is known, it is possible to read absolute values from the individual contour lines.

Scale Mapping There are several different ways to manage the way positive and negative data are treated in the scaling transformations in *map3d*. The current version supports the simplest, or *true* mapping, in which the data are used as they are with no consideration of positive or negative values—the color map spreads evenly across the range of the extrema. Subsequent versions will support the *symmetric* scale mapping, which sets the positive and negative extrema symmetrically—the larger (in the absolute value sense) determines both maximum and minimum data values. Also to appear in the net version is the *separate* scale mapping, in which the positive and negative extrema are treated completely separately—'half' the colours (and contours) are used for the positive values, half for the negative values. This is equivalent to producing maps with the same number of contours for both positive and negative values, even when the positive data have a different absolute maximum value than the negative data.

Colour Map There are four different colour maps presently implemented with every chance of more to come. The user can select which colour map to use. The choices currently implemented are:

Jet map (default) The Jet map is the same as the one used in MATLAB. Colours range from dark blue (for negative extrema) through greens (near zero) to dark red (positive extrema). Jet utilizes a minimal set of similar color, particularly of greens and yellows, a complaint of the Rainbow map.

Rainbow map Colours range in rainbow fashion from blue (for negative extrema) through greens (near zero) to red (positive extrema).

Red (+) to Green (-) Largest negative value is coloured bright green, dark grays are for the region near zero, and positive values appear red.

Black (+) to White(-) Grey shades from black for small values to white for large ones.

Note that for each color map, the direction of the mapping to value can be inverted, *e.g.,* in the default directions, blue indicates small or negative values and red indicates large or positive values. Inverted, this map uses red for small values and blue for large values.

Contours Contours will be spaced according to the scaling parameters mentioned above. The number of contours or contour spacing can be changed in the Contour Dialog (See Section 8.5.5). If 'contour spacing' is selected, the spacing will determine the number of contours based on the range. If the function is linear and the mapping is true, the gap between contours will be the number specified in the dialog.

7.4.2 Scalar data reference

Related to scaling is the reference channel used for the displayed scalar data. By default, we assume that scalar values already have the right reference and we do nothing to change that. The user can, however, select a new reference and then subtract that reference from all signals in the surface. This is done by selecting the "Reference lead - single value" or "Reference lead - mean value" from the Picking menu (See Section 8.6). There are at present two types of references that *map3d* supports:

Mean as reference: Selecting the mean as reference causes *map3d* to subtract the average value over each surface for each instant in time from the scalar data on that surface. Selecting the "Reference lead - mean value" from the Picking menu automatically does this, and can be undone by selecting "Reset Reference" from the same menu.

Selected lead as reference: It is also possible to select a single channel of data and use that as the reference signal. This is done by first entering the the pick mode called "Reference lead - single value", and then selecting the reference node (See Section 8.2.1) performs this operation. The rest of the nodes then use that node as a reference value. Selecting a new reference lead works properly, *i.e.,* the effect is not cumulative but first restores the data to the original state, than applies the new reference, and this can all be undone by selecting "Reset Reference" from the same menu.

7.5 Landmarks

Landmarks provide a means to include visual icons and markers in the surface display in *map3d*. They are not meant to render realistically but simply to be cues to assist the user in identifying perspective or features of the surfaces. The list of support landmarks reflects our current usage for bioelectric field data from the heart but many of the landmark types are general purpose and hence useful in other contexts.

Section 6.5 describes the currently support landmark types and the files that contain them. Display of each landmark type depends on the type and user controlled options (see Section 8 for details on controlling the display).

7.6 Clipping Planes

Clipping planes allow you to remove from view certain parts of the display so that you can better see what is left. So everything on one side of the clipping plane is visible and everything on the other is not.

We have two clipping planes in *map3d* and their position and alignments are adjustable as well as their relation to each other—we can lock the clipping planes together so they work like a data slicer, always showing a slice of constant thickness.

The controls for clipping planes are adjustable from the menus (see Section 8.2.3) and also via keyboard controls (see Section 8.2.2. The basic controls turn the two clipping planes on and off, lock them together, and lock their position relative to the objects in the surface display. By unlocking the last control, you can select that part of the display you want to clip; the default clipping planes are along the z-axis of the object (up and down). To control position of the planes along their normal direction, just keep hitting the bracket keys ([] and {}).

7.7 Background image

A background image may be useful in providing additional information. See the usage. If you want the image to line up in the geometry, you can specify geometry coordinates to force the image to line up with the geometry by specifying opposite coordinates of the geometry as parameters to the image. Note that the z in the (x,y,z) min and the (x,y,z) max should be the same.

7.8 Node marking

Node markings are just additional information added to the display of the nodes. This may be as simple as drawing spheres at the nodes to make them more visible, or as elaborate as marking each node with its associated scalar data value. Section 8.2.3 describes these options in detail.

7.9 Time signal display

Display option for the time signal are very modest in this version of *map3d*. This will change. . .

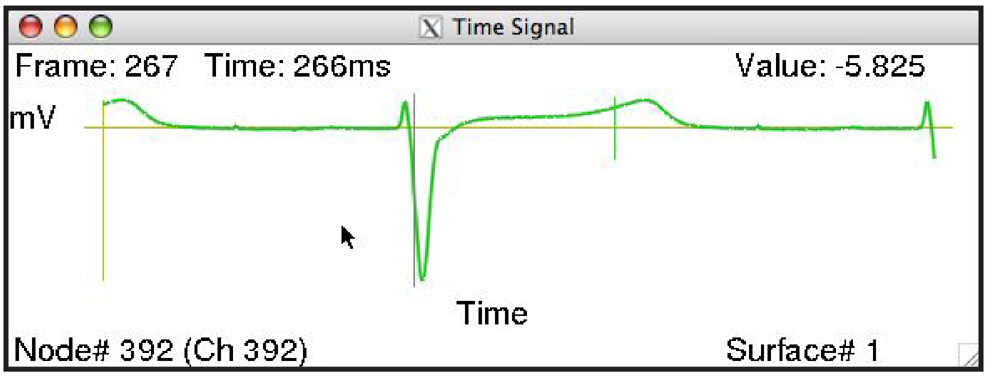

Figure 2: Time signal window layout. Vertical line indicates the frame currently displayed in the surface plot. Text annotations can vary with the data content and mode settings.

Figure 2 shows the layout and labeling of the scalar window. Font sizes adjust with the window size and the type of units may be explicit if the time series data (`.tsdf`) files contain this information.

For directions on how to control the time signal window, see Section 8.4.

8 Control of *map3d*

This section describes all the means of controlling the function of *map3d*, at least all the ones we are willing to tell you about.

8.1 Control by surface

There is an ever growing number of parameters that the user can alter for displaying the surfaces in *map3d*. Some of the more important (and stable) include the following:

Visibility: of points, mesh, potentials, vectors, *etc,* can all be controlled individually by using the appropriate function key (see Section 8.2.2),

Lead markings: of the nodes in the geometry according to their node number, channel number, lead number or even value,

Landmarks: appearance of the landmarks on the surface.

Since this level of control is provided for each surface, it is possible to have points showing on one surface, mesh on another, and rendered potential shading on a third, and so on.

8.1.1 Selecting which surface to control

To control the display of each surface, be that a surface in its own window or sharing a single window with other surfaces, a user must select that surface. Otherwise, display options will affect all the surfaces. There are two different multi-surface situations and each has its own method of selecting the surface:

Selecting surfaces for display controls	
Multi-surface layout	Selection method
One surface per window	Mouse location establishes currently active window.
Several surfaces in one window	Up/down arrows selects the surface. Hitting the up-arrow key after selecting the last surface selects all surface.

Note that in the surface window, the lock icons in the lower left corner indicate if parameter settings act on all surface (locks visible) or just single surfaces (locks invisible). Each lock controls a different aspect of *map3d*:

General Lock: is represented by the yellow (or first) lock icon. When this lock is active, menu items and keyboard commands pertain to all surfaces. To turn this lock off and on use the up and down arrow keys.

Transformation Lock: is represented by the red (or second) lock icon. When this lock is active, rotation and translation pertain to all surfaces. To turn this lock on and off, use the "t" key.

Frame Lock: is represented by the blue (or third) lock icon. When this lock is active, frame advancing, retreating, and resetting pertain to all surfaces. To turn this lock on and off, use the "f" key or select from the menu under *Frame Controls*.

Note: in the case where all surfaces are in the same window, unlocking the general lock by means of the up or down arrow keys selects the single surface to display. However, when the general lock is active and either of the other locks is disabled, the active surface mesh appears in a different color (blue by default). This identifies the selected surface and all modifications apply to this surface. To select the desired surface use the (/) keys; "(" selects the next surface and ")" selects the previous.

8.2 Mouse control, keyboard mapping, dials, and menus

Direct interactive control of *map3d* is by the keyboard and mouse. Many option are available via the menus controlled with the right mouse button, while others can be activated or toggled with single keystrokes. Variable (non-binary) adjustments usually occur through dialogues, or by repeating keystrokes. Below are tables of all the current control devices and their function. When the program launches, the user sets one or more windows which can be resized and moved at any time. When launching the program with the `-b` option, the resulting borderless sub-window(s) can still be moved and resized within a main window using the Alt-key together with the left and middle mouse buttons respectively. In Mac OS X and other operating systems where the Alt-key is mapped to another function you may use the CRTL+SHIFT keys as an alternative to the Alt-key.

8.2.1 Mouse control

The mouse can be used for different purposes. Figure 3 shows the various actions of the mouse buttons.

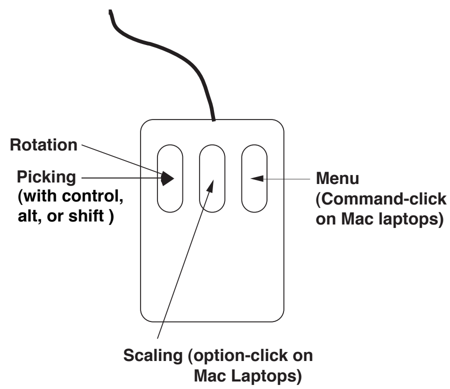

Figure 3: Mouse action for *map3d*. Picking makes intensive use of the mouse, as does moving objects in the surface window.

In surface windows: when the mouse is over a surface window, mouse buttons have the following actions:

Mouse Actions		
Control Key	Button	Action
None	Left	rotation objects
	Middle	scale objects (downwards increases size, upwards decreases size)
	Right	activate pull-down menu
Cntrl	Left	pick a node (and if time series data is present, select the channel to display in the time series window)
	Middle	no action
	Right	no action
Shift	Left	translate objects
	Middle	scale objects (rotates clipping planes if they are active - more info later)
	Right	no action

In borderless windows: when the mouse is over a surface within a borderless main window (-b option), the buttons have the following additional actions:

Mouse Actions		
Control Key	Button	Action
Alt(Ctrl+Shift)	Left	Move a single surface subwindow
	Middle	Resize single surface subwindow (no indication of change until release of mouse button).
	Right	no action

Note: if *map3d* does not respond as described in these tables, it may be that your window manager is grabbing the mouse/key combinations for its own purpose or maps the keys a little differently. This will require some setting changes for the window manager. In Linux there is usually a control panel or utility application to manage all window system interactions.

8.2.2 Keyboard controls

Each key of the regular keyboard, the function keys, and the keypad may be mapped to some function of the *map3d*. Some keyboard keys serve as toggles to change between a mode being on or off, *e.g.,* "n" toggles the display of node markings. Others cycle through a set of choices, *e.g.,* "m" runs through a series of display options for the mesh. Several lists of the keyboard keys and their functions are shown in Tables 1–4.

Table 1: Keyboard controls for Geometry Window in map. When control contains both lower and upper cases of a letter, one cycles through a parameter in one direction and the other in the reverse direction.

Regular keyboard	
a/A-key	Switch colour tables
c-key	Toggle contour draw (also, when fiducial display is active, toggles betwen fiducial contours and scalar data contours
d-key	Toggle depth cuing
f-key	Toggle frame lock (also in Time signal window)
g-key	Toggle style of Gouraud shading (between texture-mapped and non-texture-mapped)
i-key	Toggle "direction" (invert) of color table
l-key	Toggle use of lighting
m/M-key	Step through mesh/node drawing options
n-key	Toggle display of node labels (some node marking must be selected)
p-key	Toggles whether surface shading affects scalar data map or fiducial map
q-key	Quit or destroy a sub-window - Legend Window and Time Signal Window only (Escape quits the whole program)
r-key	Reset to startup conditions
R-key	Reset shading model (to wireframe rendering)
s/S-key	Cycle through the various surface data draw options (affecting either the scalar data map or the fiducial map - see p-key)
t-key	Toggle transformation lock
w-key	Write an image to a file. This will append a 4-digit number representing a image sequence number to the base filename (before the extension). The base filename can be set at start-time with the -if flag (see Section 4.2), or in the Image/Animation Control Dialog (Section 8.3.2).
x-key	Draw axis
Escape	Quit the program, if pressed in a Geometry Window, or Destroys a Time Signal Window if pressed there.
+/- key	Increases/Decreases size of of currently-selected object (see Section 8.5.3)
(&) keys	Change which surface inside a window will be affected when the general lock is on but the transform or frame lock is off (see Section 8.1.1)

Table 2: Clipping plane controls

Clipping Controls	
<-key	Toggle front clipping plane
>-key	Toggle rear clipping plane
[&] keys	Move front clipping plane in (initially) +z/-z direction respectively
{ & } keys	Move rear clipping plane in (initially) +z/-z direction respectively
,-key	Lock/Unlock clipping plane rotation with object rotation (when unlocked, shift-Middle-click rotates clipping planes)
.-key	Lock/Unlock clipping planes from each other. When active, clipping planes move together

Table 3: Control of *map3d* via the arrow keys

Arrow Keys	
Left Arrow Key	Retreat by current frame step (Also in Time Signal Window)
Right Arrow Key	Advance by current frame step (Also in Time Signal Window)
Up Arrow key	Select next surface
Down Arrow Key	Select previous surface

Table 4: Keypad controls in *map3d* - have NUM-Lock off for these to work properly. Again, based on how you have your keys mapped, you might have to use the non-keypad keys, but something should work for you for each key.

Keypad Keys	
Ctrl-Keypad Left-arrow	Y-axis rotate, CW (left)
Ctrl-Keypad Right-arrow	Y-axis rotate, CCW (right)
Ctrl-Keypad Up-arrow	X-axis rotate, CCW (down)
Ctrl-Keypad Down-arrow	X-axis rotate, CW (up)
Ctrl-Keypad Home	Z-axis rotate, CCW
Ctrl-Keypad PgUp	Z-axis rotate, CW
Ctrl-Keypad End	Zoom down
Ctrl-Keypad PgDn	Zoom up
Alt(or CRTL+SHIFT)-Keypad Left-Arrow	-X-translation (left)
Alt(or CRTL+SHIFT)-Keypad Right-Arrow	+X-translation (right)
Alt(or CRTL+SHIFT)-Keypad Down-Arrow	-Y-translation (down)
Alt(or CRTL+SHIFT)-Keypad Up-Arrow	+Y-translation (up)
Alt(or CRTL+SHIFT)-Keypad Home	-Z-translation (away)
Alt(or CRTL+SHIFT)-Keypad PgUp	+Z-translation (towards)
Plus/Minus key	Increase/Decrease size of currently-selected object. (see Section 8.5.3)

8.2.3 Menu layout

Access to the menus is by means of the right mouse button, as per the usual OpenGL convention. Below is a series of tables of the menu layout for *map3d*'s Geometry Window.

Table 5: The overall menu structure in the Geometry Window

.

Overview of Geometry Window menus	
Files	Opens the Files Window (see Section 8.3.1)
Save	Various saving options
Contours	number or spacing and display features of the contours
Fiducials	Fiducial display features - Currently only opens the fiducial dialog. (See Section 8.5.6)
Frame Controls	modifying frame controls
Graphics	general display features such as lighting, clipping, and depth cuing
Landmarks	features for toggling and displaying landmarks
Mesh	display features of the mesh
Node marking	marking of the nodes
Picking	selecting times signals, mesh information, or other direct interactions with the display via the mouse
Scaling	links between data values and color
Surface Data	display features of the scalar data displayed on the mesh
Use +/- to select	Select a feature to change interactively with + or -
Window Attributes	features of the windows such as color and text labels

Table 6: Menus for saving options.

Save Menu		
Image/Animations		Opens the Save Window (see Section 8.3.2)
Settings		Saves a .map3drc file from the currently used settings. This file goes in your home directory nad loads each time you run *map3d*, so it will behave similarly each time.
Script file		Saves a script file (bash) that loads all the files and applies all the settings to recreate the current view (see Section 5)
Windows batch file		Saves a windows batch file that loads all the files and applies all the settings to recreate the current view (see Section 5)

Table 7: Menus for contour spacing/number.

Contour Menus		
Set Contour Num/Spacing		Opens a dialog to change user-specified spacing or number of contours. See Section 8.5.5.
Draw style		
	Dashed line for negative values	draw positive contours in solid, negative in broken lines
	Solid lines for all contours	draw all contours in solid lines
Line size		set the line thickness
Toggle contours		toggle display of contours without changing settings

Table 8: Frame Controls.

Frame Control Menus		
Lock Frames		toggle whether frames operations affect one surface or all surfaces
Set Frame Interval		
	Set Frame Step	Opens a dialog to interactively set User-specified frame step. Affected surfaces are determined by the lock status.
	User-specified Interval	use the value of interval specified in the `-i` option of the command line or the value set in the frame step dialog.
	value	select between 1 and 90 for frame animation step
Reset Frames to 0		positions the surface at the first position in time
Align meshes to this frame num		Positions all surfaces' frames to the current surface. What 'current surface' means will vary based on the status of the locks (see Section 8.1.1)
Set time to zero		Set current frame to be time zero.

Table 9: Graphics menus. These control general graphic rendering options.

Graphics Menus		
Light source		select the source for the lighting model
	From above	
	From below	
	From left	
	From right	
	From front	
	From back	
	None	no lighting (turn lighting off)
Toggle clipping		toggles particular clipping plane options
	Front plane	toggles front clipping plane
	Back plane	toggles rear clipping plane
	Locking planes together	makes planes translate together
	Locking planes with object	rotate surface with the planes or rotate surface through the planes
Toggle Depth cue		apply/disable depth cuing (fog)
Adjust Depth cue		Opens a dialog to adjust the front and rear planes to where the depth cuing occurs.

Table 10: Submenus for the Mesh Display menu

Mesh render menu		
Render as		
	None	Do not render the mesh
	Elements	render filled surfaces for all elements
	Connectivity	render connectivity mesh
	Elements and connectivity	rendered front facing triangles as elements and back facing as connectivity
	Points	render points
	Points and connectivity	render both points and connectivity
Line/point size	set the size of the mesh's points and lines	
Color	set the color of the mesh	
Secondary Mesh Color	set color of active mesh when multiple surfaces are in same window	
Show Legend Window	display mesh's legend window	
Hide Legend Window	turn off display of mesh's legend window	
Reload Geometry	reload file associated with this geometry	
Reload Surface Data	reload file associated with this geometry's surface data	
Reload Both Geometry and Data	reload files associated with this geometry and its data	

Table 11: Menus for marking nodes in the display. If all surfaces are currently displayed, any of these settings will affect all surfaces based on the rules in locks section (see Section 8.5.3). If we have a single or current) surface only, then change only that surface.

Node Marking Menus		
All		Make all the nodes in this (or all) surface(s)
	Sphere	mark each node with a sphere
	Map data to spheres	mark each node with a sphere, whose color reflects its scalar data value
	Node #	mark each node with the node number in the geometry
	Channel #	mark each node with the associated data channel number
	Data value	mark each node with the associated data value
	Color	set the color for marking all nodes
	Size	set the size of all node markings
	Clear all marks	remove all node marking settings
Extrema		Make all the nodes that are the extrema
	Sphere	mark each extrema with a sphere
	Node #	mark each extrema with the node number in the geometry
	Channel #	mark each extrema with the associated data channel number
	Data value	mark each extrema with the associated data value
	Size	set the size of all extrema markings
	Clear all marks	remove all extrema marking settings
Time signal		Make all the nodes that identify the location of time signals shown in the display
	Sphere	mark each times signal location with a sphere
	Node #	mark each times signal location with the node number in the geometry
	Channel #	mark each times signal location with the associated data channel number
	Data value	mark each times signal location with the associated data value
	Color	set the color for marking all times signal locations
	Size	set the size of all time signal markings
	Clear all marks	remove all time signal marking settings
Lead links		Make all the nodes that identify the features from leadlinks file (see Section 6.4) shown in the display
	Sphere	mark each lead location with a sphere
	Node #	mark each lead location with the node number in the geometry
	Channel #	mark each lead location with the associated data channel number
	Data value	mark each lead location with the associated data value
	Lead labels	mark each lead with the label from the leadlinks file.
	Color	set the color for marking all lead locations
	Size	set the size of all lead markings
	Clear all marks	remove all lead marking settings

Table 12: Pick mode menus, part 1. Picking is based on a mode (selectable in this menu), and is done with CTRL-left mouse button unless otherwise specified.

Picking Menus I	
Time Signal (new window mode)	create a new time signal window with each pick of a node
Time Signal (refresh window mode)	update the last time signal window with each pick of a node
Display node info mode	Picking will cause certain node information to be dumped to the console.
Display triangle info mode	Picking will cause information about the triangle you click in to be dumped to the console. It may be easier to pick triangles with clipping planes on.
Triangle construction/deletion	Normal picking (ctrl-clicking on nodes) will select points to form a triangle (triangulate). Clicking the first two nodes in this fashion will display the selected nodes, and then a line between the two. When the third is clicked, a new triangle is displayed and added to the geometry. CTRL-middle clicking selects a triangle (and not nodes) to be deleted. Again, it may be easier to pick triangles with clipping planes on.
Flip triangle mode	Will change the order of drawing the triangle's points. This will cause front-facing triangles to become back-facing and vice-versa. See Section 7.3, mesh rendering options for information on how to tell which way a triangle is facing.
Edit node mode	Will allow you to pick a point and translate it with the keyboard transform controls (see Section 8.2.2, keypad controls)
Edit landmark point mode	Will allow you to pick a landmark point and translate it with the keyboard transform controls (see Section 8.2.2, keypad controls)
Delete node mode	Will remove from the geometry any node that you pick and any triangles associated with it.

Table 13: Pick mode menus, part 2. Picking is based on a mode (selectable in this menu), and is done with CTRL-left mouse button unless otherwise specified.

Picking Menus II	
Reference lead, single value	Causes all values to be measured against the node which you pick
Reference lead, mean value	Causes all values to be measured against a mean value of all the nodes
Reset Reference	Causes the reference and values to be reset to its original value. You must reset the reference if you want to change the reference more than once.
Show Time Signal Window info	Toggles display modes in time series window. When there is no info, the graph takes up more room in the window. This is equivalent to pressing 'p' in a geometry window or selecting the toggle display mode option of a time series window's menu.
Show all pick windows	Causes all Time Series (Pick) Windows associated with this geometry to become visible.
Hide all pick windows	Causes all Time Series (Pick) Windows associated with this geometry to become hidden.
Size of picking aperture	select the size of the region around the mouse pointer that will register a "hit" when picking; larger values will make it easier to pick an object but also easier to hit multiple objects.
Size of triangulation node mark	when triangulating, a node mark will appear on the node(s) selected. Adjust that mark's size.

Table 14: Menu for scaling, the mapping from data value to color for rendering.

Scaling Menus		
Scaling...	opens up the scaling dialog. (see Section 8.5.4)	
Range		
	Local	scale based on the local extrema for each surface and time instant
	Global over all frames in one surface	scale based on the extrema over the full times series
	Global over all surfaces in one frame	scale based on the extrema over each surface for the local time instant
	Global over all surfaces and frames	scale based on the extrema over all surfaces and all time instants
	Scaling over groups in one frame	scale based on the extrema over each surface in specified group for the local time instant
	Scaling over groups in all frames	scale based on the extrema over each surface in a specified group for all time instants
	Slave Scaling over one frame	scale based on the extrema over each surface's master surface (set with -sl in command line) for one time instant
	Slave Scaling over all frames	scale based on the extrema over each surface's master surface (set with -sl in command line) for all time instants
Function		
	Linear	linear mapping between value and color
	Logarithmic	color changes as log(value)
	Exponential	color changes as exp(value)
	Lab standard	color reflects lab standard
	Lab 13 standard	color reflects lab 13 standard
Mapping		
	True	use true extrema
	Symmetric about zero	take largest of absolute values of extrema to determine scaling
	Symmetric about midpoint	scale independently on both sides of the midpoint to determine scaling
	Separate about zero	scale positive and negative portions of the scale independently
Grouping		
	Move to group #	Select a group to place the current surface (make sure the general (yellow) lock is off, or all surfaces will be placed in that group)

Table 15: Submenus to control the display of scalar data on the mesh.

Surface Display Menus		
Color		
	Rainbow	use rainbow color map to render scalar values on the mesh
	Green to red	use green to red color map
	White to Black	use black and white color map
	Invert	invert the sense of any color map, *e.g.,* black becomes white and white becomes black
Render style		
	None	Turn Shading off
	Flat	colour each mesh element in a constant color according to the mean value of scalar data over the vertices
	Gouraud	shade each polygon using linear interpolation
	Banded	draw the regions between contour lines as bands of constant color

Table 16: The Use +/- to interactively change sizes menu

+/- Adjust Menu.	
Large Font Size	Change the largest font size (This is the title for the Colormap and Geometry windows)
Medium Font Size	Change the medium font size (This is all text not small or large)
Small Font Size	Change the small font size (This is the size for the node marks, the axis labels in the Time Signal Window, and the Contour Labels in the ColorMap Window)
Contour Size	Change the contour width
Line/Point Size	Change the width of Lines/Points in the mesh
Node Marks (all) Size	Change the width of node marks
Node Marks (extrema) Size	
Node Marks (time signal) Size	
Node Marks (leads) Size	
Change in translation	Change how fast the keyboard translation happens
Change in scaling	Change how fast the keyboard scaling happens
Change in rotation	Change how fast the keyboard rotation happens

Table 17: Controls for the attributes of the *map3d* windows.

Window Attributes Menus		
Screen info		select the text written to the screen. Note that screen info disappears when clipping is on.
	Turn screen info on	
	Turn screen info off	
	show/hide lock icons	toggle lock display
	Show Legend Window	Turns on Colormap window
	Hide Legend Window	Turns off Colormap window
Color		select window colors with the separate color selector
	Background	select background color for the window
	Foreground	select foreground color for the window
Size		select some size options
	value	set window to specified resolution
Axes		select options for axes
	Axes Color	Select axes color
	Axes Placement	select whether axes displayed per window or mesh
	Toggle Axes	turn on/off axes
Font Size		Use the Size Picker to adjust the font size
	Small Font Size	Size of node mark text or Colormap window contour ticks
	Medium Font Size	Size of window subtitles or text in Pick Window
	Large Font Size	Size of Window titles
Toggle Transformation Lock		toggle whether surfaces transform together or independently

Table 18: Controls for the attributes of the Time Signal Window.

Time Signal Window Controls	
Axes Color	Select Axes Color
Graph Color	Color of data graph
Toggle Display Mode	Show basic values and graph, show detailed values and graph, or show larger graph

Table 19: Controls for the attributes of the Legend window.

Legend Window Controls		
Orientation		Layout of information in Legend Window
	Vertical	
	Horizontal	
Number of Tick Marks		select number of ticks to appear on bar
	2,4,8	Either 2,4,8 ticks. These will not be colored as they do not directly correspond to contour values.
	Match Contours	Match number of contours in corresponding geometry

Table 20: Controls for the Legend Window Menu

.

Main Window Controls	
Set Background Color	Select bg Color
Quit Map3D	Quit Map3D

8.3 Interactive GUIs - File Selection, File Saving, and Scaling Options, *etc,*

With the move to a dedicated GUI system (in our case, gtk), we make use of a number of different windows to select parameters and control *map3d*. This section covers the functionality of all of them.

8.3.1 Files Window

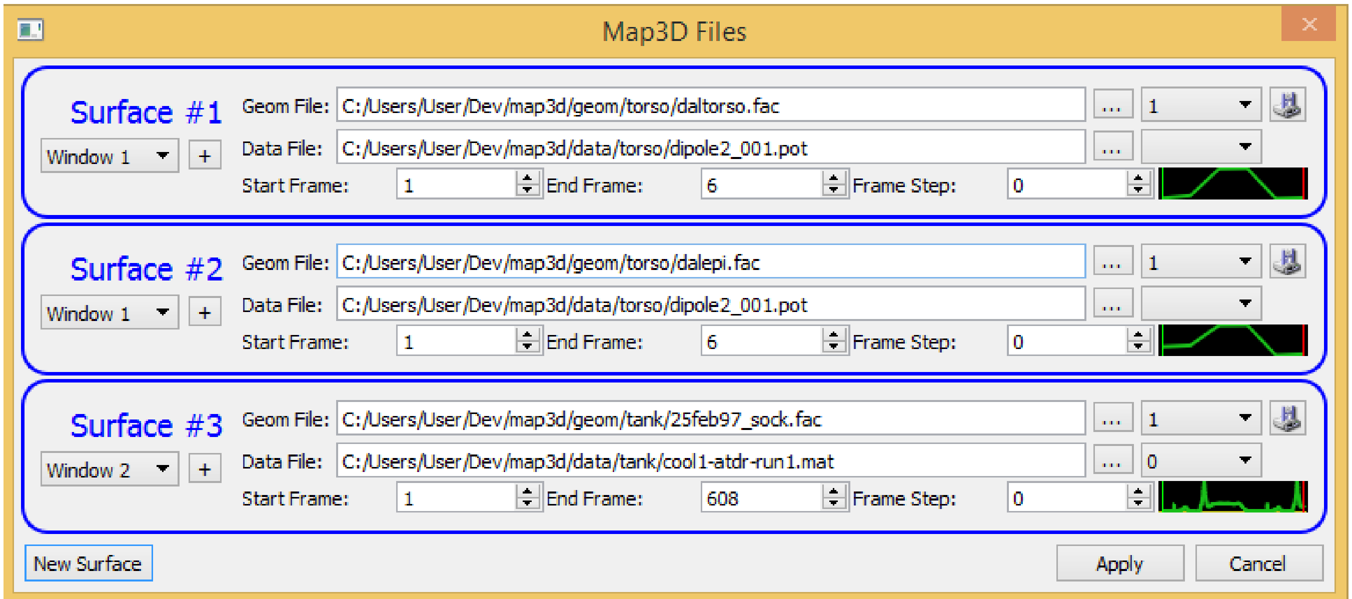

Figure 4: Files Dialog for *map3d*.

The most frequently used GUI window is the "*map3d* Files" window, allows you to interactively select filenames, data windows, etc. The others are for saving files, and for quick scaling changes. Figure 4 shows an example of such a file window, and access to this window is by means of the "Files" menu option of the main menu. This window displays one row for each surface, where each row shows the surface number, the window the surface appears in, the geometry filename, the geometry number, the data filename, the start frame number, the end frame number, a graph of the RMS curve, and a button to show other files. Most of these columns can be modified at any time. If you click on the "New Surface" button, an empty row will pop up, allowing you to add a surface from scratch.

None of the changes made to this window will take effect until you click on the "Apply" button. The "Close" button will cause the window will close, but opening it again will reveal the same content.

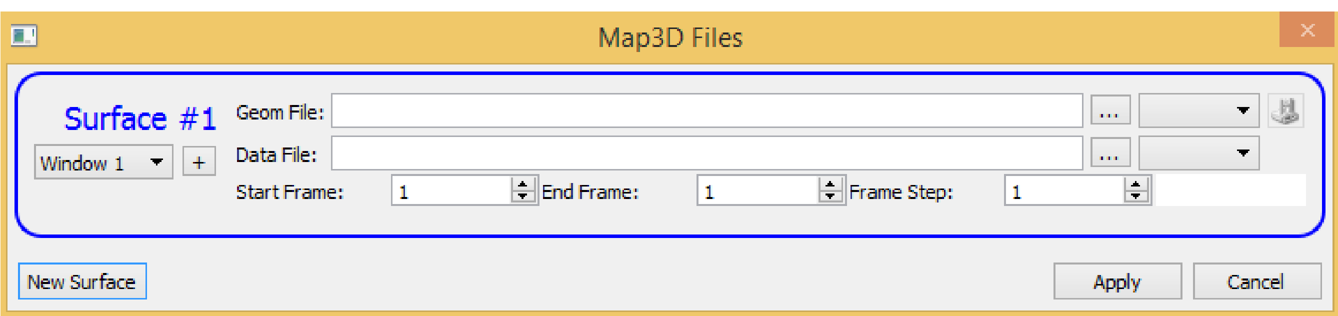

Figure 5: Empty Files Dialog. *map3d* will start up this way if launched without arguments.

Note that the File window allows launching of *map3d* without any arguments. In this case an empty file window appears, as in Figure 5 so that the user can select all the files for display and set up the display interactively.

Table 21: Options for the file window

File Window Options	
Surf	Number of the surface
Win#	Number of the window that contains this surface. You can change this number to move the surface to any open window, or to a new window (by selecting the last number from the drop-down menu).
Geom File	Name of the geometry file of this surface. To change the geom file click on the ... button and then browsing for the desired file. You may reload the current geometry by selecting the "Mesh⇒Reload Geometry" menu entry.
Geom#	Number of surface within the geometry file (see Section 6.1). If there is more than one surface in the geometry file, an associated drop-down menu selects which one to insert.
Geom Save (disk icon)	Allows saving of the current geometry (see Section 8.3.1).
Data File	Name of the data file of this surface. As with geometry clicking on the ... button launches a file browser to select a time-series data file. Selecting the "Mesh⇒Reload Data" reloads the data.
Data Selection	If more than one time series is present in the data file, the selection will allow you to choose between them
Start Frame	Start reading data at this frame. Left-clicking in the graph section and dragging will also select the correct frame.
End Frame	Finish reading data at this frame. Middle-clicking in the graph section and dragging will select the correct frame.
Frame Step	If larger than 1, *map3d* will load every nth frame
Graph	Graph of the root-mean-square signal calculated from all signals in the time series. Left- or middle-clicking in this graph selects the data window that *map3d* will read and display.
Other Files (+)	This button will expand another section in which to view/modify the channels, leadlinks, landmarks, or fiducial file to use with this surface.

Saving Geometry Make sure to modify the filename or the current filename will be replaced. Files can be named with .mat, .fac, or .pts extensions. If the filename ends in the extension .fac, it will save filename.fac and filename.pts.

8.3.2 Saving Images/Animations

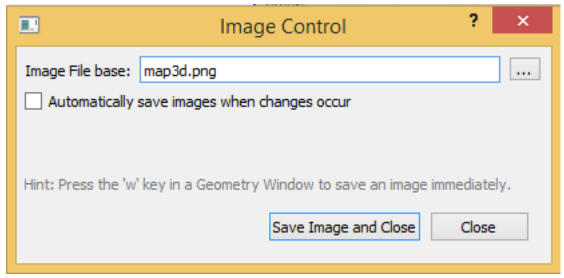

Figure 6: Image Control Dialog.

Saving Images The Image Control dialog is used to control the base filename for saved images and whether to save animations. Select a filename (you can click on ... to browse for a filename) and click save. The filename may have the extension .ppm, ,png, or .jpg to save in one of those formats. The final filename also appends a 4-digit number before the extension, representing a number in a sequence of images. I.e., if the filename selected is map3d.png, the image will actually save in map3d0000.png, and subsequent images will be map3d0001.png, map3d0002.png, etc. The image that will saved is approximately the smallest rectangle that contains all open map3d windows.

NOTE: If there are windows that overlap, the one on top might not be the one *map3d* thinks is on top. So if after moving windows around, it is necessary to click inside the window (not the title bar), to tell *map3d* it is on the top. Otherwise, data that appears on the computer screen may end up obscured by other windows that appear to lie beneath.

If the Save... window is in the way, close it, and pressing the 'w' key will have the same effect as clicking the "Save Image" button.

Saving Animations Checking the "Automatically save image when changes occur" box allows you to control automatic frame saving to put images together into movies. While *map3d* is not yet sophisticated enough to actually create the movies, this control can save the images you need and then you can use some external software to make a movie from them. See Section 9.1.2 for links to instructions of how to make movies from sets of images.

There are three changes types that determine whether an image will be saved.

1. Save Frame on Transformation - will save a frame (image) every time you transform (rotate, translate, scale) any surface, either with the mouse or the keyboard.

2. Save Frame on Frame Advance - will save a frame when you move forward or backward in time with the arrow key, or change the time in the pick window.

3. Save Frame on Other Events - will save a frame when you interact with with the Geometry Window in any other way, via keyboard commands or menu controls.

Naturally, animations start recording when you check the box, and stop when you uncheck the box.

8.4 Controlling the time signal window

There are two ways to create a time signal window:

1. Specify a `-at xmin xmin ymin ymax` on the command line (optionally with a `-t trace-lead-number` to specify the channel to use).

2. Using picking (Cntrl-left mouse) to select a lead to show in the time signal window, when the current pick mode is time signal new-window mode or time signal refresh mode. Note that subsequent time signal picking can be set to either a) update the last time signal window to the new data channel or b) add yet another time signal window.

The format of the scalar display is fairly simple , with a vertical bar moving along the time axis as the frame number is advanced. *map3d* derives the time axis label from the frame numbers of the signal relative to the time series data file, not relative to the subset of frame read in, *i.e.,* if frames (or pot file numbers) 10–20 are read in with an increment of 2, then frame number will begin at 10, and go through 12, 14, 16, 18 and end at 20 rather then beginning at 0 or 1 and going to 10 (the number of frames of data actually read).

8.4.1 Adjusting the frame selector

In order to facilitate rapid movement through large datasets, the user can control the frame number being displayed by interacting with the scalar window itself. If the user moves the cursor to the scalar window and pushes the left mouse button, the vertical time bar will jump to the nearest sample to the cursor location. The user can then hold the left button down and slide the time marker left and right and set a desired frame. Once the mouse button is released,*map3d* updates the map display. The left and right arrow keys also shift the frame marker back and forth. The only other command allowed when the cursor is within the scalar window is the “q”-key, to shut down just the scalar window, the “f” key, to toggle the frame lock, or the “p” key, to toggle the display mode. Any other attempt at input will not be accepted.

8.5 Color/Size Selection

It is frequently necessary to control color and size of elements of the *map3d* display and this, we have selection subwindows that appear as necessary and disappear upon selection.

8.5.1 Color Picker

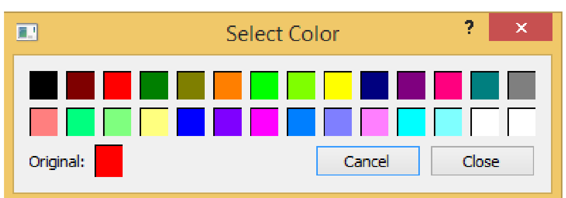

Figure 7: Color Picker.

The Color Picker shows a number of colors to select from. There are only a limited number of colors so that the color selection can be easily reproduced on subsequent runs. When you open up the Color Picker, the current (original) color will be in a small box on the bottom left, whereas the box next to it will show the color that was most recently selected. There is a new “Preview” button which will change the color and let you see what it would do, but will also allow you to push the “Cancel” button and return to the original color, shown in the bottom left.

8.5.2 Size Picker

The Size Picker shows 10 sizes to select from, currently (to change later) represented by the size of boxes, where the width of the box represents the selectable size. When you open up the Size Picker, the current (original) size will be represented in a box on the bottom left, whereas the box next to it will show the size that was most recently selected. There is a new “Preview” button which will change the size and let you see what it would do, but will also allow you to push the “Cancel” button and return to the original size, shown in the bottom left.

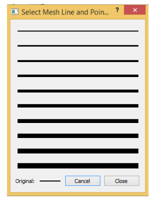

Figure 8: Size Picker.

8.5.3 Interactive Size Control

Rather than having to open the Size Picker over and over, *map3d* provides a few options that can be changed by a single keystroke. To do this, open the menu in the Geometry Window and select "Mesh-¿Use +/- to select" and then select a feature you wish to dynamically adjust. Then press + or - to adjust the size. However, most of these only have 10 possible sizes.

8.5.4 Scaling Options

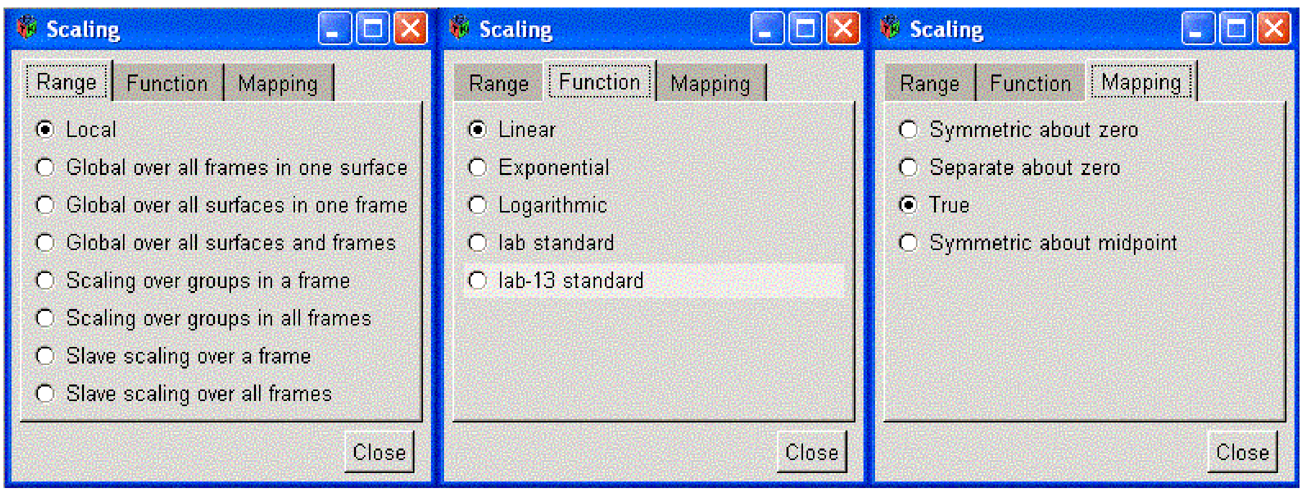

Figure 9: The three tabs of the Scaling Dialog.

This three-tabbed dialog contains the same options as the Scaling Menu in the Geometry menu, but in dialog form for convenience. Click on the tab at the top of the scaling type you wish to change (Range, Function, or Mapping), and click on the check box of the feature you wish to select, and *map3d* will update. (see Section 8.2.3, scaling menu) and Section 7.4.1.

8.5.5 Contour Setting Dialog

The contour setting dialog allows selection of the contour spacing, number of contours and scaling range. A side-effect of this dialog is that a change of one feature has consequences on other features. For example, changing the number of contours will likely also change the spacing between the contours. Alternatively, it could change the range of the scaling, *i.e.,* the maximum and minimum used to map value to color. This interdependence of the control parameters can make this dialog somewhat obscure but it does allow broad flexibility of settings.

There are two controls that affect the interplay among features - when the range changes, adjust either: Contour Spacing or Num Contours. The choice between these two will determine which parameter will remain constant when the range changes (the range can change by selecting a different contour or by setting the range explicitly).

A second, related control is the "Fixed Range" check box on each row. If it is checked, then changing the contour spacing will affect the number of contours and vice-versa. If it is checked, then changing the number of contours or the spacing will affect the range, and changing the range will affect either the

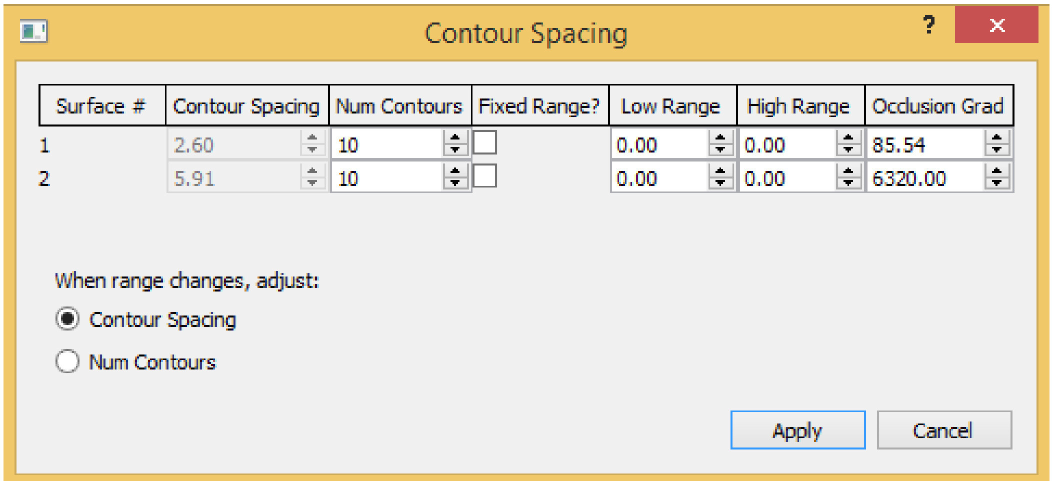

Figure 10: Contour Spacing Dialog.

number of contours or the contour spacing. Also, unselecting "Fixed Range" will set the scaling range for this surface to the range specified in the contour dialog and will not reflect the default range parameter.

The occlusion gradient is used to ignore data on triangles whose gradient exceeds the number specified. This is useful if there would be too many contours present. The default number is the total range of the entire dataset, in effect disabling the feature.

Upon pressing "Apply", the changes will take effect. Note that the exact contour spacing will only take effect if the scaling function is set to Linear and the mapping is set to True. Otherwise, an approximate contour spacing will result.

This description may sound confusing but some trial and error should hopefully provide complete control over the display of scaling features. Complexity is the cost of flexibility and when in doubt, we have always tended toward flexibility.

8.5.6 Fiducial Control

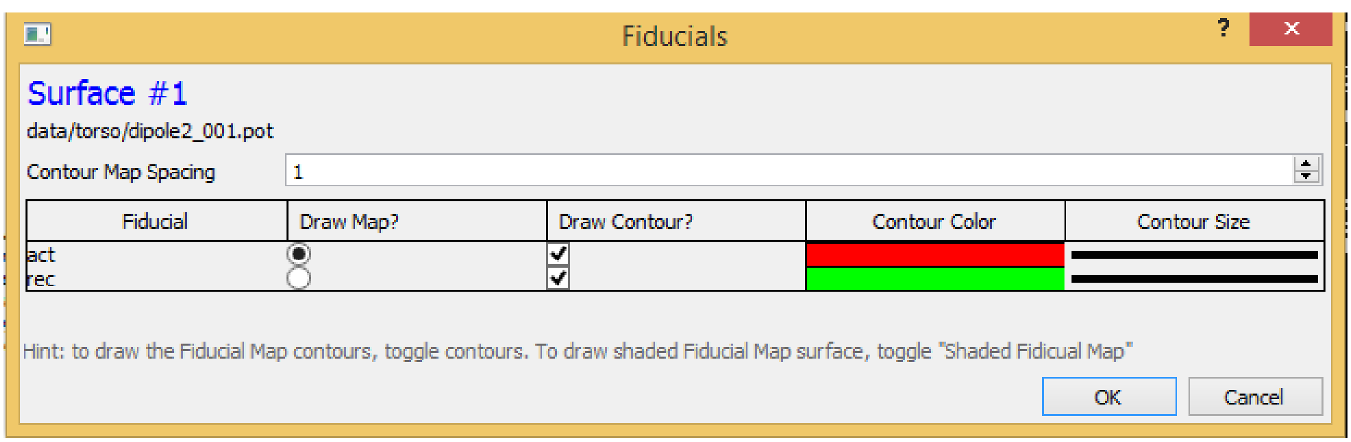

Figure 11: The Fiducials Control Windows.

Fiducials are time markers in the signals, for example, indicating the time of activation or the start and end of the QRS complex in an ECG. Because the main application domain of *map3d* is cardiac electrophysiology, the selection of support fiducials is related to signals from the heart. However, there is not reason one could not add new fiducials (contact Rob MacLeod (macleod@cvrti.utah.edu))

There are two ways of viewing fiducials.

1) Single fiducial contour: in this mode, *map3d* draws an isochrone line over top of the potential map that corresponds to the time value of the current potential map. This sounds complicated but it means that, for example, if the potential map has a time value within the range of activation times for the time series, then somewhere on the map will be an isochrone corresponding to that activation time. In practice, one sees the activation contour expand with each step in time and can also perform quality control of any fiducial detection algorithm

The lower window in Figure 11 allows control of the display of the contour, setting the contour thickness and the color associated with each time of fiducial.

2) Multiple fiducial contours or maps: This is the more traditional model of fiducial map display in which the entire set of isochrone values maps up a map. To enable simultaneous viewing of potential and fiducials in this mode, *map3d* provides a number of options controlled by the "p"-key, and the "s"-key, and the other options in the Fiducials menu. With various combinations, it is possible to display a) potentials as shaded maps with black isochrones, b) no potentials with colored isochrones, and c) potentials as black contours drawn over shaded color isochrone map. The window allows selection of the particular fiducial to display.

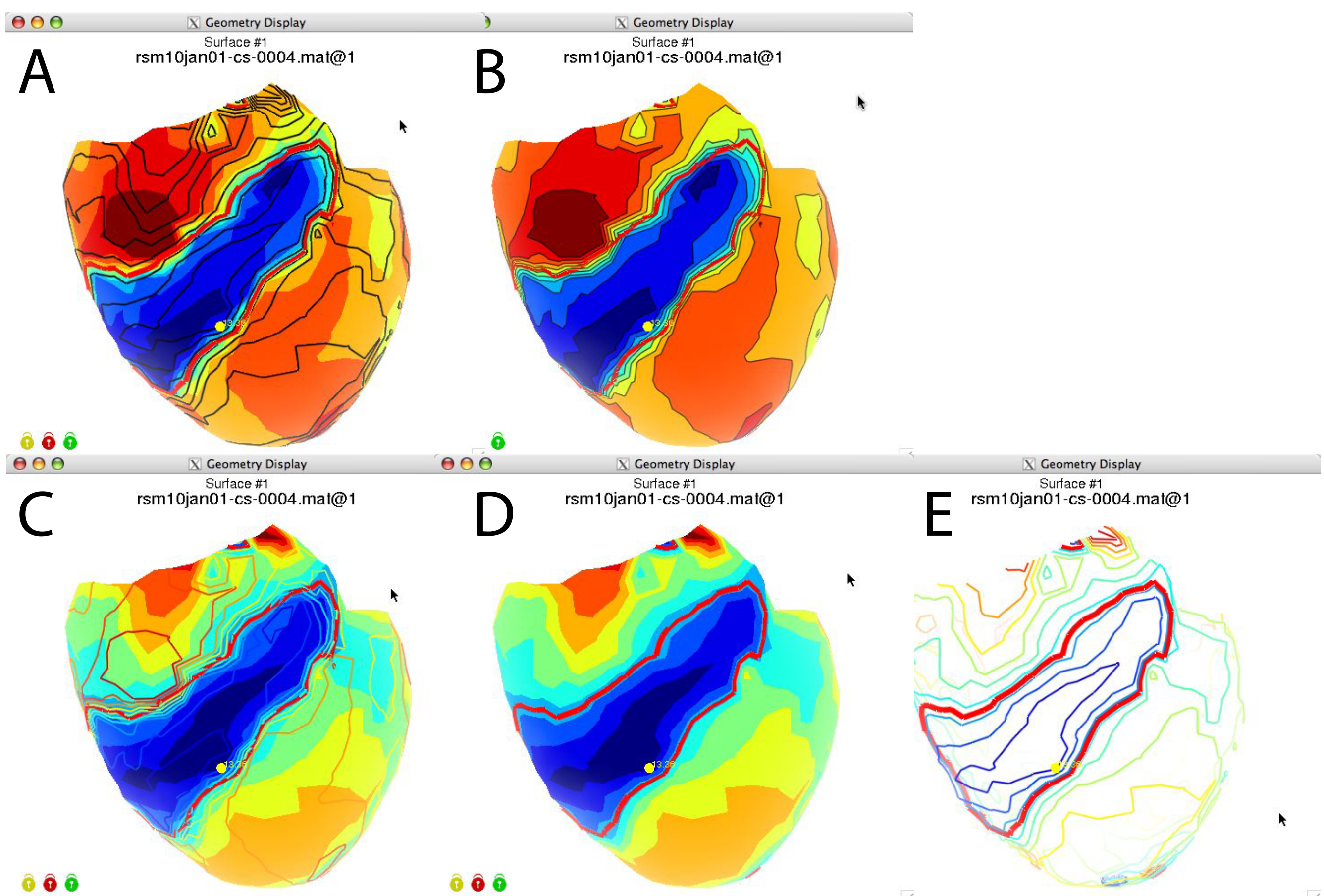

Figure 12: Examples of fiducial displays. Panels A, and B show different forms of activation contour display over shaded potential maps. Panels C and D show displays of shaded activation maps with superimposed potentials (Panel C) and a single activation contour (Panel D). Panel E shows a contour display of activation time.

Figure 12 shows different forms of display of activation times using the fiducial display options. The choice of which is most informative depends on the perspective of the user and the problem at hand. The images in Panel A and D are useful to perform a quick quality control of that fiducial map. Panel B supports a more detailed analysis by allowing the user to step through time and see fiducial contours associated with each time value superimposed on the potentials. In this way, the user can use spatial information to evaluate the quality of the fiducial values. For a more temporal vie w, the user can click on any number of time signals (see Figure 2) and see the fiducial markers superimposed as vertical lines that are color coded according to fiducial type.

8.6 Picking mode

By "picking" we mean selecting some piece of the display in the current window using the mouse. *map3d* currently supports selection of nodes or triangles with different actions, all of which either return some information, affect the display, or even alter the geometry of the display. The current choices include node information, triangle information, time signal displays, triangulation, triangle deletion, triangle flipping, node editing/deletion, landmark editing. Note that picking is successful and the desired results occur only when there is one (no more, no less) hit. To aid in getting the one hit, you may adjust the picking aperture or activate the clipping planes. In any picking mode where your geometry is modified, you might want to save your geometry (see Section 8.3.2). To control picking, use the top-level "Picking" menu. See Tables 12 and 13 for the available options.

Time Signal (new window mode): This mode opens up a new window and provides information about the node/channel. It tells the node number, channel number, the associated surface, the current frame number, and the value of the scalar data at the current time instance. In addition, it shows a graph of the scalar data associated with that node over time.

Time Signal (refresh window mode): This mode provides the same information and graph as the new window mode, but displays the information in the most-recently created window (and creates one if one doesn't exist).

Node Information: This mode outputs information of the selected node to the console: Surface number, frame number, node number, channel number, current data value, and the X,Y,Z coordinates of the point (as read by the geometry file).

Triangle Information: This mode outputs information of the selected triangle to the console: Surface number, triangle number, and the numbers and X,Y,Z coordinates of the 3 triangle points (as read by the geometry file).

Triangle Construction (Triangulating) and Deletion: In this mode, the left mouse is used to select the nodes that you wish connected into a triangle. Note that each time you select a valid point a change is made to the geometry. The first valid point you will see a new point on the mesh. The second will be similar except there will be a line connecting the two points, and the third time will add the completed triangle to the geometry

Pushing CTRL-middle mouse button selects a triangle and kills it, removing it from the list.

Triangle flipping: Often it is necessary to know the orientation of the triangles in the geometry. While this can be computed, there remains a 180-degree ambiguity as to which way the normal points. To resolve this, triangles nodes should be ordered in a counterclockwise direction as viewed from the "outside" of the surface to which the triangle belongs. This convention is used by OpenGL to decide which triangles to show in "hide back-facing triangles" mode. Unfortunately, it is not always possible when constructing geometric models to tell which way the triangle is to be viewed—this is still something humans do better than computers—and so we often need to edit a geometric model so that the triangles are 'flipped' the right way. Hence, the "Flip triangle" option in the "Picking" menu.

Edit node: If your surface isn't perfect, you can edit the positions of the nodes. Select this pick mode in the Picking menu, and pick a node. A mark will appear on the node that you pick. Then use the keyboard transformation controls to translate a node (see Section 8.2.2, keypad controls). Note that keyboard rotation won't work in this mode.

Edit landmark point: If your landmarks aren't perfect, you can edit their positions. Select this pick mode in the Picking menu, and pick a landmark point. A mark will appear on the point that you pick. Then use the keyboard transformation controls to translate it (see Section 8.2.2, keypad controls). Note that keyboard rotation won't work in this mode.

Delete node: In this mode, any node you pick and any triangle that connects with it will be removed from the geometry.

Reference lead, single value: In this mode, the value of the node you pick will be used as the reference from which the other nodes will be measured.

Reference lead, mean value: In this mode, a mean value of all the nodes will be used as the reference from which the nodes will be measured.

8.7 Control of landmark display

Table 22: Landmark menus. These control landmark display options.

Landmarks Menus		
Coronary/Catheter		toggle and select color for coronary/catheter
	Toggle Coronary	toggle coronary artery
	Toggle Catheter	toggle catheter display
	Wireframe Coronary	show coronary artery as wireframe and pt. numbers
	Coronary Color	select coronary color
	Catheter Color	select catheter color
Points		toggle and select color for point landmarks
	Toggle temporary occlusions	toggle temporary occlusion display
	Toggle permanent occlusions (stitch)	toggle stitch display
	Toggle stimulation site	toggle stim display
	Toggle recording site (lead)	toggle lead display
	Occlus Color	select occlus color
	Stitch Color	select stitch color
	Stim Color	select stim color
	Lead Color	select lead color
	Toggle All Points	turn on/off point landmarks
Planes		toggle and select color for plane landmarks
	Toggle Plane	toggle plane display
	Toggle Plane Transparency	make plane transparent or opaque
	Plane Color	select plane color
Points		toggle and select color for point landmarks
	Toggle rod	toggle rod display
	Toggle recording needle	toggle recneedle display
	Toggle pace needle	toggle pace needle display
	Toggle fiber (lead)	toggle fiber display
	Toggle cannula (lead)	toggle cannula display
	Rod Color	select rod color
	Recneedle Color	select recneedle color
	Paceneedle Color	select paceneedle color
	Fiber Color	select fiber color
	Cannula Color	select cannula color
	Toggle All Rods	turn on/off rod-type landmarks
Toggle All Landmarks	turn all landmarks on/off	

There are some lighting and colour controls for the display of landmarks that are useful to know about. Table 22 describes all the specific menus mentioned here.

Coronary/Catheter: these are controls to adjust color and visibility of the vessel type landmark, the ones we use for arteries and also for catheters. You can also switch from a rendered version of this landmark type to a wire mesh that is labeled according to segment numbers—a debugging tool when the vessels are not going where you expect.

Points: there are a range of point types in the landmark suite and this option allows you to control them turn them off and on, adjust color, etc.

Planes: for the plane landmark, you can set color, but also the transparency level of the plane. Default is transparent and this usually works best.

Rods: as with points, there are different rod type landmarks and with this menu you can control their visibility and color.

Toggle all landmarks: this is the master switch and toggling it turns all the landmarks from on to off or vice versa.

9 Output from *map3d*

9.1 Capturing images for animation, printing, or photos/slides

While screen images are lovely to look at, we need to be able to get the output from the screen to some transportable medium like paper, animation movies, video tape, or film. This section describes some of the methods available for this process.

9.1.1 Image capture

There are no standard provisions in OpenGL for generating output from the images generated by *map3d*. However, *map3d* uses a collection of the GL windows to create an image and save it to a file. Once preserved, this file can be viewed later, either by itself or as part of a sequence of images in an animation.

To capture an image using *map3d* simply set the image you want to preserve and hit the "w"-key. There will be a slight pause and the a line will appear in the control window telling you where the image has been stored. Filenames for image storage are generated automatically, using the filename specified in the Saving dialog, which defaults to the value set with the `-if` option or it will default to map3d.png (See Section 8.3.2). Appended to this base filename are sets of four digits, denoting the frame number currently in the display, starting with "0001". Thus, for example, if the base image file were daltorso.png, the first file produced would be `daltorso0001.png`. Note the `.png` file extension, standard for this sort of file, can also be changed to .ppm or .jpg.

The screen area captured in this mode is the smallest rectangle that contains all the windows currently managed by the current invocation of *map3d*. This often requires with careful placement of the windows or setting the background window for the display to black or something that matches the background of the *map3d* display.

9.1.2 Animations

Sometimes it is desirable to save a sequence of images in a movie for use in a demonstration. *map3d* does not (currently) have the ability to save movies directly, but it does have the ability to automatically save a sequence of images based on a set of input events, which can be pieced into a movie from external software. The images are saved into a sequence of files based on the rules in the image capture section, and each time the appended digits increment. See Section 8.3.2 for more information on how to control the animations.

Making movies There are a few commercial programs we have found useful in generating movies directly:

1. Snapz Pro, which is a marvelous program from Ambrosia Software for grabbing frames in real time from the screen.
2. Final Cut Pro, a program from Apple that is as good as most professional tools (so they say).
3. iMovie HD, which comes free on a Mac. It is worth checking the iLife version, which is usually a little ahead the version shipped with Macs, if you are serious about editing video.

Otherwise, while we are working on integrating movie support directly into *map3d*, there are a few packages to create movies from your frames.

1. QuickTimePro (for Mac OSX), if you want this one in a version that allows editing, you'll have to pay.
2. Snapz Pro (for Mac OSX), also some cost involved but very modest for what this program will do.
3. Discreet Cleaner XL (for Win32).
4. mencoder (for Linux or Windows).

5. ffmpeg is a cross-platform utility you can use to generate movies. However, you would have to download it and compile it yourself. Once you have downloaded and compiled it, you can, for example: ffmpeg -i map3d%04d.jpg map3d.mpg, which will turn map3d0001.jpg, etc. into map3d.mpg.

We are still learning which combinations of settings work best to capture, edit, and save animations. It depends a lot on the context in which you plan to view/show the results. As we learn more, we will share it with you.

References

[1] C.R. Johnson and R. S. MacLeod. High performance computing in medicine: Direct and inverse problems in cardiology. In *Proceedings of the IEEE Engineering in Medicine and Biology Society 15th Annual International Conference*, pages 582–583. IEEE Press, 1993.

[2] C.R. Johnson and R. S. MacLeod. Nonuniform spatial mesh adaptation using a posteriori error estimates: applications to forward and inverse problems. *Applied Numerical Mathematics*, 14:311–326, 1994.

[3] C.R. Johnson and R. S. MacLeod. Local regularization and adaptive methods for the inverse Laplace problem. In D. N. Ghista, editor, *Biomedical and Life Physics*, pages 224–234. Vieweg-Verlag, Braunschweig, 1996.

[4] C.R. Johnson, R.S. MacLeod, and M.A. Matheson. Computational medicine: Bioelectric field problems. *IEEE COMPUTER*, pages 59–67, October 1993.

[5] R.L. Lux, M. Akhtar, and R.S. MacLeod. Mapping and invasive analysis. In P.M. Spooner and M.R. Rosen, editors, *Foundations of Cardiac Arrhythmias: Basic Concepts and Clinical Approaches*, chapter 15, pages 393–424. Marcel Dekker, 2001.

[6] R.L. Lux, P.R. Ershler, and B. Taccardi. Measuring spatial waves of repolarization in canine ventricles using high resolution epicardial mapping. *J. Electrocardiol.*, 29(Suppl):130–134, 1996.

[7] R.S. MacLeod and D.H. Brooks. Recent progress in inverse problems in electrocardiology. *IEEE Eng. in Med. & Biol. Soc. Magazine*, 17(1):73–83, January 1998.

[8] R.S. MacLeod, D.H. Brooks, H. On, H. Krim, R.L. Lux, and F. Kornreich. Analysis of PTCA-induced ischemia using both an electrocardiographic inverse solution and the wavelet transform. *J. Electrocardiol.*, 27(Suppl):90–96, 1994.

[9] R.S. MacLeod, P.R. Ershler, C.R. Johnson, and M.A. Matheson. Map3d: Scientific visualization program for multichannel time series data on unstructured, three-dimensional meshes. program user's guide. Technical Report UUCS-94-016, University of Utah, Department of Computer Science, 1994.

[10] R.S. MacLeod and C.R. Johnson. Map3d: Interactive scientific visualization for bioengineering data. In *Proceedings of the IEEE Engineering in Medicine and Biology Society 15th Annual International Conference*, pages 30–31. IEEE Press, 1993. http://software.sci.utah.edu/map3d.html.

[11] R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization of cardiac bioelectricity — a case study. In *Proceedings of the IEEE Visualization 92*, pages 411–418. IEEE CS Press, 1992.

[12] R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization tools for computational electrocardiography. In *Visualization in Biomedical Computing*, pages 433–444, Bellingham, Wash., 1992. Proceedings of the SPIE #1808.

[13] R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualizing bioelectric fields. *IEEE Comp. Graph. & Applic.*, 13(4):10–12, 1993.

[14] R.S. MacLeod, R.L. Lux, and B. Taccardi. A possible mechanism for electrocardiographically silent changes in cardiac repolarization. *J. Electrocardiol.*, 30(Suppl):114–121, 1997.

[15] R.S. MacLeod, Q. Ni, B. Punske, P.R. Ershler, B. Yilmaz, and B. Taccardi. Effects of heart position on the body-surface ECG. *J. Electrocardiol.*, 33((supp)):229–238, 2000.

[16] R.S. MacLeod, B. Punske, S. Shome, B. Yilmaz, and B. Taccardi. The role of heart rate and coronary flow during myocardial ischemia. *J. Electrocardiol.*, pages 43–51, 2001.

[17] R.S. MacLeod, B. Taccardi, and R.L. Lux. The influence of torso inhomogeneities on epicardial potentials. In *IEEE Computers in Cardiology*, pages 793–796. IEEE Computer Society, 1994.

[18] R.S. MacLeod, B. Taccardi, and R.L. Lux. Electrocardiographic mapping in a realistic torso tank preparation. In *Proceedings of the IEEE Engineering in Medicine and Biology Society 17th Annual International Conference*, pages 245–246. IEEE Press, 1995.

[19] Q. Ni, R.S. MacLeod, and R.L. Lux. Three-dimensional activation mapping in canine ventricles: Interpolation and approximation of activation times. *Annal. Biomed. Eng.*, 27(5):617–626, 1999.

[20] Q. Ni, R.S. MacLeod, R.L. Lux, and B. Taccardi. A novel interpolation method for electric potential fields in the heart during excitation. *Annal. Biomed. Eng.*, 26(4):597–607, 1998.

[21] Q. Ni, R.S. MacLeod, B.B. Punske, and B. Taccardi. Computing and visualizing electric potentials and current pathways in the thorax. *J. Electrocardiol.*, 33(Suppl):189–198, 2000.

[22] B.B. Punske, W.E. Cascio, C. Engle, H.T. Nagle, L.S. Gettes, and T.A. Johnson. Quantitative characterization of epicardial wave fronts during regional ischemia and elevated extracellular potassium ion concentration. *Ann Biomed Eng*, 26(6):1010–1021, November-December 1998.

[23] B.B. Punske, R.L. Lux, R.S. MacLeod, M.S. FUller, P.E. Ershler, T.J. Dustman, Y. Vyhmeister, and B. Taccardi. Mechanisms of the spatial distribution of QT intervals on the epicardial and body surfaces. *J. Cardiovasc. Electrophysiol.*, 10(12):1605–1618, 1999.

[24] B.B. Punske, Q. Ni, R.L. Lux, R.S. MacLeod, P.R. Ershler, T.J. Dustman, M.J. Allison, and B. Taccardi. Spatial methods of epicardial activation time determination. *Annal. Biomed. Eng.*, 31:781–792, 2003.

[25] Y. Serinağaoğlu, R.S. MacLeod, B. Yilmaz, and D.H. Brooks. Epicardial mapping from venous catheter measurements, body surface potential maps, and an electrocardiographic inverse solution. *J. Electrocardiol.*, 35(supp):65–74, 2002.

[26] B. Taccardi, R.L. Lux, P.R. Ershler, R.S. MacLeod, and Y. Vyhmeister. Effect of myocardial fiber direction on 3-D shape of excitation wavefronts and associated potential distributions in ventricular walls. *Circ.*, 86(Suppl):I–752, 1992.

[27] B. Taccardi, R.L. Lux, P.R. Ershler, R.S. MacLeod, C. Zabawa, and Y. Vyhmeister. Potential distributions and excitation time maps recorded with high spatial resolution from the entire ventricular surface of exposed dog hearts. In *IEEE Computers in Cardiology*, pages 1–4. IEEE Press, 1992.

[28] B. Taccardi, R.L. Lux, R.S. MacLeod, P.R. Ershler, T.J. Dustman, and N. Ingebrigtsen. Assessment of spatial resolution of body surface potentials maps in localizing ventricular tachycardia foci. *J. Electrocardiol.*, 30(Suppl):1–4, 1997.

[29] B. Taccardi, R.L. Lux, R.S. MacLeod, P.R. Ershler, T.J. Dustman, M. Scott, Y. Vyhmeister, and N. Ingebrigtsen. ECG waveforms and cardiac electric sources. *J. Electrocardiol.*, 29(Suppl):98–100, 1996.

[30] B. Taccardi, R.L. Lux, R.S. MacLeod, P.R. Ershler, T.J. Dustman, and Y. Vhymeister. ECG waveforms and cardiac electric sources. *J. Electrocardiol.*, 29(Suppl):98–100, 1996.

[31] B. Taccardi, E. Macchi, R.L. Lux, P.R. Ershler, S. Spaggiari, S. Baruffi, and Y. Vyhmeister. Effect of myocardial fiber direction on epicardial potentials. *Circ.*, 90:3076–3090, 1994.