map3d User’s Guide

Version 6.3
Last update: May 19, 2005

Authors: Rob MacLeod (macleod@cvrti.utah.edu),
        Bryan Worthen (worthen@sci.utah.edu), and
        J.R. Blackham (blackham@sci.utah.edu)

Scientific and Computing Institute (SCI)
www.sci.utah.edu
Center for Bioelectric Field Modeling, Simulation and Visualization
www.sci.utah.edu/ncrr
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 The Nora
Eccles Treadwell Foundation
## Contents

1 What’s New?  .................................................. 4  
  1.1 Version 6.3: May, 2005  ........................................... 4  
  1.2 In the works (“vapourware”)  .................................... 4  

2 Introduction ..................................................... 5  
  2.1 Acknowledgments  .................................................. 5  

3 Installation ..................................................... 7  
  3.1 System requirements  .............................................. 7  
  3.2 General Installation  .............................................. 8  
  3.3 Linux Installation Instructions  ................................... 8  
  3.4 Windows Installation Instructions  ................................ 9  
  3.5 Mac OS X Installation Instructions  ................................ 9  
  3.6 SGI Installation Instructions  ................................... 10  
  3.7 Installing from source  ............................................ 11  
  3.8 Documentation  ................................................... 13  
  3.9 Bug reporting  ...................................................... 13  
  3.10 How to reach us  .................................................. 13  

4 Usage ............................................................ 14  
  4.1 Typical usage examples  .......................................... 14  
  4.2 Global Parameters  ................................................ 16  
  4.3 Geometry specifications  ........................................... 16  
  4.4 Scalar Data parameters  .......................................... 18  

5 Script Files ....................................................... 19  
  5.1 What are script files?  ............................................ 20  
  5.2 How to make script files  ......................................... 20  
  5.3 Examples  ......................................................... 21  

6 Input files ....................................................... 24  
  6.1 Geometry input files  ............................................. 24  
  6.2 Command line control of geometry files  ......................... 25  
  6.3 Scalar data files  ................................................ 26  
  6.4 Channels and leadlinks  ......................................... 28  
  6.5 Landmark files  .................................................. 31  

7 Display features ................................................ 33  
  7.1 Multiple surfaces  ............................................... 33  
  7.2 Surface display  .................................................. 33  
  7.3 Mesh Rendering  ................................................... 33  
  7.4 Surface Data Display  ............................................ 33  
  7.5 Landmarks  ....................................................... 36  
  7.6 Clipping Planes  .................................................. 36  
  7.7 Node marking  ..................................................... 36  
  7.8 Time signal display  ............................................. 36
8 Control of map3d

8.1 Control by surface .......................................................... 37
8.2 Mouse control, keyboard mapping, dials, and menus ................. 38
8.3 Controlling the time signal window ..................................... 39
8.4 Color/Size Selection ......................................................... 43
8.5 Interactive GUIs - File Selection, File Saving, and Scaling Options, etc, .... 46
8.6 Picking mode ................................................................. 55
8.7 Control of landmark display ............................................. 56

9 Output from map3d

9.1 Capturing images for animation, printing, or photos/slides ............ 57

10 BUGS ........................................................................ 59
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 6.3: May, 2005

This is the fourth version of the “new” map3d with a GTK-based GUI. We are getting very very very close to the complete functionality of the old GL based version and have gone well beyond it in some features, especially the user interface. This is a “dot” release but is not a minor release for it contains some important new features and the usual set of bug fixes.

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

**Open Source:** map3d is now open source! See Section 3.7 and Section 2.1.1.

**Jet Color map:** map3d has seen the light and added the Matlab Jet color map as the default color map.

**Fiducial display:** Some improvements over the fiducial display and control. (LINK) See fid dialog, display, Section 6.3.4.

**Geometry in landmark form:** Per the recommendation of a user, we have incorporated the landmark file format into an actual geometry See Section 6.1.5.

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

1.2 In the works (“vapourware”)

A small sampler of things that are in the works:

- Incorporating fiducials into the matlab format
- Better fiducial control
- 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 10).
2 Introduction

This document describes the function and usage of version version 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 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 \(^1\)–\(^4\) and a technical report \(^5\) and we’d love it if you would reference at least one of them (perhaps \(^3\) or \(^4\) 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.\(^3\)–\(^30\)

One of the big changes in version 6.3 is that we are now completely open source. People can download not only the executable 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 obvious interest is when someone ports 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 introduction to GL and C and this program became my personal sandbox 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 and this newest version 6.3 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 sandbox architecture, pulled together the essential walls, created new ways to make rooms, and still left lots of the sandbox 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. Most recently, J.R. Blackham has joined the team while still an undergraduate in Computer Science at Utah. 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
2.1 Acknowledgments

CVRTI/SCI Manual: map3d

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 Geometric Modeling, Simulation, and Visualization in Bioelectric Field Problems (www.sci.utah.edu/ncrr).

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 sandbox. The Nora Eccles Treadwell Foundation has also provided support for the development of map3d and the huge pile of data we have used it to analyze.

Rob MacLeod, May 19, 2005.

2.1.1 Open Source License

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

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

1. The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
2. Use of this software in preparing any publication material must be cited as follows:


THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.1.2 Libraries used by map3d

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

- GTK - The GIMP Toolkit - Copyright (C) 1995-1997 Peter Mattis, Spencer Kimball and Josh MacDonald
- GtkGLExt - GtkGLExt - OpenGL Extension to GTK+ Copyright (C) 2002-2004 Naofumi Yasufuku
- PNG - Copyright (c) 1998-2002 Glenn Randers-Pehrson
- JpegLib - Copyright (C) 1991-1998, Thomas G. Lane.

We use GTK and GtkGLExt 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 four of these libraries are covered by the GNU LGPL, which is included in the distribution of map3d.

As of version 6.3, we also release internal libraries under the same license as above for the rest of map3d.
3 Installation

3.1 System requirements

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

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

<table>
<thead>
<tr>
<th>Requirements for all systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL libraries (GL and GLU)</td>
<td>version 1.1 + ¹</td>
</tr>
<tr>
<td>OpenGL/window interface library (GLX)</td>
<td>version 2.0+</td>
</tr>
</tbody>
</table>

**Linux (i386):** *map3d* requires the OpenGL library, which is available as the mesa library at www.mesa3d.org for any Linux platform. For better performance, graphics cards from companies such as nVidia (www.nvidia.com) usually provide OpenGL libraries.

<table>
<thead>
<tr>
<th>Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>kernel 2.2.x</td>
</tr>
<tr>
<td>Architecture</td>
<td>i386 (+ maybe PPC)</td>
</tr>
<tr>
<td>Applications Binary Interface</td>
<td>libc2.1</td>
</tr>
</tbody>
</table>

**Recommendations**

<table>
<thead>
<tr>
<th>Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Window system</td>
<td>XFree86 version 4.0 +</td>
</tr>
<tr>
<td>Hardware</td>
<td>3D graphics card (nVidia, 3dfx, ati)</td>
</tr>
<tr>
<td></td>
<td>128 MB main memory</td>
</tr>
</tbody>
</table>

**Windows:**

<table>
<thead>
<tr>
<th>Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>W2K/NT4.0/9x</td>
</tr>
<tr>
<td>Architecture</td>
<td>i386</td>
</tr>
<tr>
<td>Applications Binary Interface</td>
<td>win32</td>
</tr>
</tbody>
</table>

**Recommendations**

<table>
<thead>
<tr>
<th>Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>3D graphics card (nVidia, 3dfx, ati)</td>
</tr>
<tr>
<td></td>
<td>128 MB main memory</td>
</tr>
</tbody>
</table>

**Mac OS X:**

<table>
<thead>
<tr>
<th>Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Mac OS 10.3(Panther)</td>
</tr>
<tr>
<td>Architecture</td>
<td>PPC</td>
</tr>
</tbody>
</table>

**Recommendations**

<table>
<thead>
<tr>
<th>Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>3D graphics card (nVidia, 3dfx, ati)</td>
</tr>
<tr>
<td></td>
<td>256 MB main memory</td>
</tr>
</tbody>
</table>

**SGI:** *map3d* runs on virtually any SGI that will support Irix version 6.5+. We have tested it on Indigo2, O2, Octane, and Origin workstations running various flavors of Irix 6.5. If you need a version for a different SGI configuration, please let us know (map3d@sci.utah.edu)
3.2 General Installation

Unfortunately, with our move to GTK+ for window support, it is not as easy as past versions, which required just the download of an executable. We hope (in vain, perhaps) to be able to do that again in the future, 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](http://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

The Linux installation is relatively straightforward. You’ll need to download `map3d`’s dependencies, then download `map3d` itself.

3.3.1 Linux Dependencies

There are two phases to this part. First we need to get GTK+ and its dependencies. The easiest way to do this is from your distribution’s installation CDs, or you can download the RPMs at [www.rpmfind.net](http://www.rpmfind.net).

To get the dependencies from your distribution, run the Package Manager (Add/Remove Applications, configure-packages or something of that sort). Search for gtk, and install gtk2 (if you can’t find that directly, then installing the gnome environment will take care of it).

To get the dependencies from the internet, navigate your favorite browser to [http://www.rpmfind.net](http://www.rpmfind.net) and search for gtk-2.0. Try to find one that matches your distribution (redhat, mandrake, etc.). We directly support development for gtk2-2.2.1 and gtk2-2.0.6, so if you can find one of these that would be encouraged.

The next part is to download gtkglext, the library that supports OpenGL for GTK widgets. As of this release, this is not standard in most distributions. If your GTK version is 2.0.6 or 2.2.1 (you can find out by looking at `gtkversion.h` which will be where you installed gtk (normally `/usr/include/gtk-2.0/gtk/gtkversion.h`), and look for GTK_MAJOR_VERSION, GTK_MINOR_VERSION, and GTK_MICRO_VERSION. There will be numbers on the same lines as each of these, and if you put them together it will be something like 2.2.1). If you are using one of these versions download `gtkglext-linux-2.2.1.tar.gz` or `gtkglext-linux-2.0.6.tar.gz` from the `map3d` download page and follow these instructions:

```sh
cd <download directory>
gunzip gtkglext-linux-<version>.tar.gz
tar xf gtkglext-linux-<version>.tar
cp libgtkglext-x11-1.0.so.0 /usr/local/lib
cp libgdkglext-x11-1.0.so.0 /usr/local/lib
```

You can copy them to some directory other than `/usr/local/lib` if you wish.

If this doesn’t work, you will need to download the gtkglext source and compile it yourself (don’t worry—if your gtk is properly set up, this will be very easy). Download the sources from Source Forge [http://sourceforge.net/projects/gtkglext](http://sourceforge.net/projects/gtkglext) and follow these instructions:
3.4 Windows Installation Instructions

```bash
cd <download directory>
gunzip gtkglext-1.0.6.tar.gz
tar xf gtkglext-1.0.6.tar
cd gtkglext-1.0.6
configure
make
make install
```

If you don’t want these to end up in `/usr/local/lib`, you need to

```bash
configure --prefix=<dir>
```

where `dir` is where to put the libraries.

### 3.3.2 Linux Executable

Download the `map3d-6.2-linux.tar.gz` file from the `map3d()` download page and unzip it to a directory of your choice. We will call that RUN-DIR. This is the directory from which you will run `map3d`.

To run `map3d`, you will need to make sure that all the libraries are in your `LD_LIBRARY_PATH` environment variable. For this we will assume that your gtk libraries are in `/usr/lib` and your gtkglext libraries are in `/usr/local/lib`. Do the following: tcsh users:

```
setenv LD_LIBRARY_PATH /usr/local/lib:$LD_LIBRARY_PATH
```

or bash users:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

you might want to put this line in your `.cshrc` or `.profile` file to avoid having to run this multiple times.

### 3.4 Windows Installation Instructions

The Windows installation is relatively straightforward. You’ll need to download `map3d`’s dependencies, then download `map3d` itself.

#### 3.4.1 Windows Dependencies

Download the `map3d-win-environment.zip` file from the `map3d` download page and unzip (using winzip, native windows xp zip file browser, or another windows zip program) it into a location of your choice - we will call that INSTALL-DIR. It will create a directory called `INSTALL-DIR\map3d`.

Add `INSTALL-DIR\map3d\lib` to your path. To do this, open the Control Panel, select System, and click on the ‘Advanced’ tab on the top of the screen. Click on the Environment Variables button. You should see a variable called `Path` or `PATH` in the System Variables section. Click on it, and select Edit. Go to the end of the line, add a semi-colon (`;`) and type `INSTALL-DIR\map3d\lib`.

#### 3.4.2 Windows Executable

There is an executable of `map3d` in the environment directory. We have also placed an executable here to facilitate future downloads, so you only have to download the environment once. If you wish, download the `map3d-6.2-windows.zip` file from the `map3d` download page and unzip it to a directory of your choice. We will call that RUN-DIR. This is the directory from which you will run `map3d`.

### 3.5 Mac OS X Installation Instructions

The Mac OS X installation is relatively straightforward. You’ll need to download `map3d`’s dependencies, then download `map3d` itself.
3.5.1 Mac OS X Dependencies

Map3d requires the gtk+2 and gtkglext libraries. You can easily install these libraries using fink.

If you do not currently have fink installed on your system you will need to go to http://fink.sourceforge.net and follow the instructions on how to install it and the gtk+2 and gtkglext libraries.

3.5.2 Mac OS X Executable

Download the map3d-6.2-mac.tar.gz file from the map3d download page and unzip it to a directory of your choice. We will call that RUN-DIR. This is the directory from which you will run map3d.

3.6 SGI Installation Instructions

The SGI installation is relatively straightforward. You’ll need to download map3d’s dependencies, then download map3d itself.

3.6.1 SGI Dependencies

There are two phases to this part. First we need to get GTK+ and its dependencies. The easiest way to do this is for SGI is to run a browser with root access and go to SGI’s freeware site at http://freeware.sgi.com. On that page near the bottom there should be a link to a prereq calculator. Click on that link, and in the Freeware Product box, select fw_gtk2+ (make sure you don’t select fw_gtk+). Submit the query. You should see a list similar to this:

```
fw_atk [relnotes] [prereqs] [install]
fw_expat [relnotes] [prereqs] [install]
fw_freetype2 [relnotes] [prereqs] [install]
fw_gettext [relnotes] [prereqs] [install]
fw_glib2 [relnotes] [prereqs] [install]
fw_libjpeg [relnotes] [prereqs] [install]
fw_libpng [relnotes] [prereqs] [install]
fw_libz [relnotes] [prereqs] [install]
fw_pango [relnotes] [prereqs] [install]
fw_tiff [relnotes] [prereqs] [install]
```

To install them, click on the install link one by one (and follow the instructions in the dialog boxes).

IMPORTANT - when you install libz - it will mention something about a security library being removed. When you install libz, allow it to do this. On the subsequent libraries, it will mention that the security package conflicts with libz, on these packages, have it continue without installing the security package.

Install them in the order:

```
gettext
expat
freetype2
atk
glib2
pango
libjpeg
libtiff
libz
libpng
```

After you’ve done all of these, click on the alphabetical link, and click on the install button that corresponds to libgtk2+-2.0.6.

If for some reason, the prereq calculator isn’t there or isn’t working, go to the alphabetical index and install the above in the order specified.
The next part is to download gtkglext, the library that supports OpenGL for GTK widgets. If your GTK version is 2.0.6 (you can find out by looking at gtkversion.h which will be where you installed gtk (normally /usr/freeware/include/gtk-2.0/gtk/gtkversion.h), and look for GTK_MAJOR_VERSION, GTK_MINOR_VERSION, and GTK_MICRO_VERSION. There will be numbers on the same lines as each of these, and if you put them together it will be something like 2.0.6). If you are using this version download gtkglext-sgi.tar.gz from the map3d download page http://www.sci.utah.edu/software/map3d.html and follow these instructions:

    cd <download directory>
    gunzip gtkglext-sgi.tar.gz
    tar xf gtkglext-sgi.tar
    cp libgtkglext-x11-1.0.so.0 /usr/local/lib
    cp libgdkglext-x11-1.0.so.0 /usr/local/lib

You can copy them to some directory other than /usr/local/lib if you wish.

If this doesn’t work, you will need to download the gtkglext source and compile it yourself (don’t worry—if your gtk is properly set up, this will be very easy). Download the sources from Source Forge http://sourceforge.net/projects/gtkglext and follow these instructions:

    cd <download directory>
    gunzip gtkglext-1.0.6.tar.gz
    tar xf gtkglext-1.0.6.tar
    cd gtkglext-1.0.6
    configure
    make
    make install

If you don’t want these to end up in /usr/local/lib, you need to

    configure --prefix=<dir>

where dir is where you want to put the libraries (The libraries will be in dir/lib).

3.6.2 SGI Executable

Download the map3d-6.2-irix.tar.gz file from the map3d download page and unzip it to a directory of your choice. We will call that RUN-DIR. This is the directory from which you will run map3d.

To run map3d, you will need to make sure that all the libraries are in your LD_LIBRARY_PATH environment variable. For this we will assume that your gtk libraries are in /usr/freeware/lib32 and your gtkglext libraries are in /usr/local/lib. Do the following:

    tcsh users:
      setenv LD_LIBRARY_PATH /usr/freeware/lib32:/usr/local/lib:$LD_LIBRARY_PATH
    or bash users:
      export LD_LIBRARY_PATH=/usr/freeware/lib32:/usr/local/lib:$LD_LIBRARY_PATH

you might want to put this line in your .cshrc or .profile file to avoid having to run this multiple times.

3.7 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
3.7 Installing from source

3.7.1 Download Source Code

You can get the map3d source code from the map3d download page at [http://www.sci.utah.edu/software/map3d.html](http://www.sci.utah.edu/software/map3d.html). Login and work your way to the map3d version 6.3 download page, and download map3d-source.tar.gz.

3.7.2 Install Dependencies

You will need to install map3d’s dependencies: gtk+, gtkglext, etc. To do this, please refer to Section 3.4, Section 3.6, Section 3.3, or Section 3.5.

If you are running on a different system, you will probably need to download and install these on your own; please refer to GTK website and GtkGLExt website.

3.7.3 Configuring map3d

map3d in addition to GTK+, map3d has other dependencies that have been developed in conjunction with map3d, which were designed to be used with and independently of map3d. The make system shipped with map3d was designed to allow users to compile these libraries independently of map3d if they so choose. However, most users will not use these libraries for any other purpose except for running 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 gtk is. The included files show samples of how this is to be done, and following are specific details.

**MS Visual Studio users**  To configure MatlabIO, open Visual Studio, load map3dtop/map3d.sln, right-click on the MatlabIO project, and select properties. Under Configuration properties, click C/C++, select General, and add the directory where zlib.h is.

To configure map3d, right-click on the map3d project, and select properties. Under Configuration properties, click C/C++, select General, and add the directory where all the gtk-based header files are (if you installed the map3d-environment from the website, they should all be in the same place). I.e., \local\map3d-environment\include. Each directory should be semi-colon delimited.

While still under map3d property pages, select Linker, and under General, add the directory where the libraries are; i.e., \local\map3d-environment\lib.

**Mac, Linux, and SGI users**  To configure both MatlabIO and map3d, it is only necessary to modify map3dtop/Makefile.incl.

To configure MatlabIO, change ZLIBINC to the directory that contains zlib.h Also change ZLIBLIB to the directory that contains the zlib library.

To configure map3d, edit map3dtop/map3d/Makefile and change GTKTOP, GTKLIB, GTKGLINC, and GTKGLLIB to appropriate values. If gtk.h is in /usr/local/include/gtk-2.0/gtk, and libgtk-x11-2.0.so (or .dylib for mac users) is in /usr/local/lib, then

GTK_INC=/usr/local/include
GTK_LIB=/usr/local/lib

Similarly, GTKGLINC and GTKGLLIB need to be set. to include the dir that contains gtkgl.h and libgtkgl-x11-1.0.so (or .dylib) respectively. (-I goes in front of the include directory, and -L goes in front of the lib directory, respectively.)

3.7.4 Compiling map3d

Compiling map3d now should be as simple as entering the map3dtop directory and running

make

Remember for map3d to run, you should probably add map3d to your path. If, when you run map3d see errors like “Cannot load library gtk-2.0.dll” or “Cannot map so libgtk-2.0.so”, then you need to add the gtk libraries to your runtime path. To do this on windows, follow the directions on the windows install page, otherwise, set the LD_LIBRARY_PATH. To do this, do the following: tcsh users:
setenv LD_LIBRARY_PATH GTK_LIB:$LD_LIBRARY_PATH

or bash users:

export LD_LIBRARY_PATH=GTK_LIB:$LD_LIBRARY_PATH

Substitute the values that GTK_LIB is set to in the Makefile.incl. You might want to put this line in your .cshrc or .profile file to avoid having to run this multiple times.

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

### 3.8 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

### 3.9 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 software.sci.utah.edu/bugzilla (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.

### 3.10 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.
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 8.5.1). 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
```

*map3d* also has other parameters it accepts that are designed for the use of a .map3drc or script file, and those parameters will be shown in Table 20.

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.geom

  The first instance will run map3d and allow you to input files interactively (see Section 8.5.1). 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 binary geometry file (.geom extension) exists that contains both nodes and connectivities. We describe all these forms of geometry files in Section 6.1.3.

- 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
4.1 Typical usage examples

- 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),

  \[ \text{map3d} -f \text{geomfilename}.fac -p \text{datafilename}.pot -ch \text{thefile}.channels \]

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

  \[ \text{map3d} -f \text{geomfilename}.fac -p \text{datafilename}.tsdf \]

  with the time series stored in a datafile described in Section 6.3.2.

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

  \[ \text{map3d} -f \text{geomfilename}.geom -p \text{datafilename}.tsdf \]

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

- A time series data file (.tsdf) 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,

  \[ \text{map3d} -f \text{geomfilename}.fac -p \text{datafilename}.tsdf -s 1 100 -i 2 \]

  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

  \[ \text{map3d} -f \text{geomfilename}.fac -p \text{mapdata} -s 1 9 \]

- 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

  \[ \text{map3d} -f \text{geomfilename}.geom@2 -p \text{datafilename}.tsdf \]

  will select geometry #2 from the file geomfilename.geom.

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

  \[ \text{map3d} -f \text{geomfile1}.fac -p \text{thedata1} -f \text{geomfile2}.fac -p \text{thedata2} \]

  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 8.5.1).
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.

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 6.1.3 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 geometry (.geom) 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.

4. binary matlab geometry (.mat) file contains both nodes and connectivity information and may also contain channel mapping. If there are multiple surfaces in the matlab file, the same restrictions apply as in the .geom files.

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:

<table>
<thead>
<tr>
<th>RGB Value</th>
<th>Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>255 0 0</td>
<td>red</td>
</tr>
<tr>
<td>0 255 0</td>
<td>green</td>
</tr>
<tr>
<td>0 0 255</td>
<td>blue</td>
</tr>
<tr>
<td>255 255 0</td>
<td>yellow</td>
</tr>
<tr>
<td>255 0 255</td>
<td>magenta</td>
</tr>
<tr>
<td>0 255 255</td>
<td>cyan</td>
</tr>
<tr>
<td>255 255 255</td>
<td>white</td>
</tr>
</tbody>
</table>

-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.
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:

- `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.

- `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. Time-series data files (.tsdf) are binary files where all time instants are stored in one file.
  3. Matlab files (.mat) are also stored in binary format. Matlab files can also contain multiple time series.
  4. Time-series container files (.tsdfc), contain references to at least one .tsdf file, along with other information about the time series. See Section 6.3.2 for more information.

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, .tsdf, .mat, or .tsdfc 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.

- `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 time series (.tsdf) data file, `map3d` will read frames `num1` to `num2` from the file.

- `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.

- `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.

- `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 6.1.5.

-ff fidfile = Ascii file containing fiducial information. Information may be specified for each node for an arbitrary set of fiducial data. See section 6.3.4.

-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.

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 file is a simple program 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 How to make script files

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:
5.3 Examples

CVRTI/SCI Manual: map3d

```
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.

5.3 Examples

Below are some sample scripts, from simple, to fairly complex:

Set the geometry, data, and window size and location

```
map3d -f ${HOME}/torso/geom/dal/daltorso.fac \
    -as 100 500 300 700 \ 
    -p ${HOME}/maprodxn/andy3/10feb95/data/cooling.tsdf \ 
    -s 1 1000
```

map3d-tank1.sh, included with this distribution

```
MAP3D=../map3d
GEOM=../geom/tank
DATA=../data/tank

$MAP3D -nw -f ${GEOM}/25feb97_sock.fac \ 
    -p ${DATA}/cool1-atdr_new.tsdf@1 -s 1 1000 \ 
    -ch ${GEOM}/sock128.channels \ 
    -f ${GEOM}/25feb97_sock_closed.geom \ 
    -p ${DATA}/cool1-atdr_new.tsdf@2 -s 1 1000\ 
    -ch ${GEOM}/sock128.channels
```

map3d-tank2.sh, included with this distribution

```
MAP3D=../map3d
GEOM=../geom/tank
DATA=../data/tank

$MAP3D -nw -f ${GEOM}/25feb97_sock.fac \ 
    -as 200 600 400 800 \ 
    -p ${DATA}/cool1-atdr_new.tsdf@1 -s 1 476 \ 
    -at 200 600 200 420 -t 9\ 
```
5.3 Examples

CVRTI/SCI Manual: map3d

map3d-torso1.sh, included with this distribution

```sh
MAP3D=../map3d
GEOM=../geom/torso
DATA=../data/torso

$MAP3D -f ${GEOM}/daltorso.geom -p ${DATA}/dipole2.tsdf -s 1 6
```

map3d-torso2.sh, included with this distribution

```sh
MAP3D=../map3d
GEOM=../geom/torso
DATA=../data/torso

$MAP3D -f ${GEOM}/daltorso_epi.geom@1
    -p ${DATA}/p2_3200_77_torso.tsdf -s 1 200
    -f ${GEOM}/daltorso_epi.geom@2
    -p ${DATA}/p2_3200_77_epi.tsdf -s 1 200
```

Set some environment variables, then layout the whole display

```sh
#!/bin/sh
# A script for the spmag 1996 article
#
#*****************************************************************************
map3d=/usr/local/bin/map3d
map3d=${ROBHOME}/gl/map3d/map3d.sh
MAP3D_DATAPATH=/scratch/bt26mar91pack/
export MAP3D_DATAPATH
echo "MAP3D_DATAPATH = $MAP3D_DATAPATH"
basedir=/u/macleod/maprodxn/plaque/26mar91
$map3d -b -nw
    -f $basedir/geom/525sock.geom
    -as 150 475 611 935
    -at 150 475 485 610 -t 237
    -p $basedir/data/pace-center.tsdf01
    -s 65 380
    -f $basedir/geom/525sock.geom
    -as 476 800 611 935
    -at 476 800 485 610 -t 237
    -p $basedir/data/pace-center.tsdf01
    -s 65 380
    -f $basedir/geom/525sock.geom
    -as 150 475 176 500
    -at 150 475 50 175 -t 237
    -p $basedir/data/pace-center.tsdf01
    -s 65 380
```

Page 22
-f $basedir/geom/525sock.geom \
-as 476 800 176 500 \
-at 476 800 50 175 -t 237 \
-p $basedir/data/pace-center.tsdf01 \
-s 65 380
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 Binary (.geom) geometry files

At the CVRTI we have developed a binary file system for efficient storage of complex geometry and associated attributes, a part of what we call the graphicsio library. Extensive documentation of this format is available from Rob MacLeod (macleod@cvrti.utah.edu) (www.cvrti.utah.edu/~macleod/docs).

Briefly, each graphicsio geometry file contains one or more sets of node locations and, optionally, connectivities for polygonal elements composed from those nodes. It is possible in graphicsio files to associate scalar, vector, and tensor values to nodes or elements, the most relevant example of which are channel pointers, stored as a set of scalar values associated with the nodes of the geometry. Each graphicsio geometry file can contain any number of sets of geometries, each with different nodes and connectivities.
6.2 Command line control of geometry files

A typical example for *map3d* would be a single .geom file that contains information from multiple surfaces that we might want to display together.

*map3d* is capable of reading surface geometry from either single surfaces or from all surfaces contained in a multi-surface geometry file. Command line arguments controls the selection, as we describe in the next section.

6.1.4 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.
   - .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:

```matlab
>> 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.

6.1.5 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* and *map3d3d* looks for *myfile.geom, 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:
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 a binary format developed at the CVRTI.

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 CVRTI data (.tsdf and .tsdfc) format files

One efficient and flexible way to store scalar values is by means of the time-series data file format developed at the CVRTI, also as part of the graphicsio library. Each time series data file (.tsdf files) holds an entire sequence, or time series of scalar data in a single file, along with some information about the contents, type, units, and global (i.e., that apply to all channels) temporal fiducials from the time series. For more details on this file format see www.cvrti.utah.edu/~macleod/docs/graphicsio.

Here are some concepts of the time series data file structure that are relevant to the different modes of operation described in this manual.

<table>
<thead>
<tr>
<th>Extension</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>look for files with the extensions .pts, .fac, .tetra, and .geom and if an incompatible set are present (e.g., both .pts and .geom), ask user which to take</td>
</tr>
<tr>
<td>.pts</td>
<td>take only the .pts file and ignore any connectivity or .geom file</td>
</tr>
<tr>
<td>.fac</td>
<td>take .pts and .fac and ignore any .geom files present.</td>
</tr>
<tr>
<td>.geom</td>
<td>take the graphicsio geometry file and ignore any others present.</td>
</tr>
<tr>
<td>.mat</td>
<td>take the MATLAB file and ignore any others present.</td>
</tr>
</tbody>
</table>
6.3 Scalar data files

Links to geometry  The links between the channels of data in the .tsdf file and the nodes of the surface[s] over which they are displayed is established via channel array information, which is available stored as associated scalars to the nodes in the geometry file (see Section 6.1) or in explicit channels files (see Section 6.4).

Frames  By frames of data, we mean instants in the data stream representing single moments in time. For each frame, there is a set of values that for a spatial distribution or map and map3d needs to know what subset of frames are to be included in the display. To explicitly specify frame numbers, use the -s and -i options described in Section 4. As an example, the command line

```
map3d -w -f geomfile.geom@1 -p datafile.tsdf -s 10 130 -i 2
```

specifies that surface 1 from the geometry file `geomfile.geom` should be used to display frames 10 to 130, taking every second frame, from run 2 of the time series data file `datafile.tsdf`.

Time series container files (tsdfc):  There is an extension to the graphicsio library that defines a container file format for a set of time series data (.tsdf) files, and can contain parameters extracted from the associated time series. These files are actually small databases and we use a modified (patched actually) version of the GNU Database Library (GDBL) to manage them.

Examples of programs and libraries that provide support for .tsdfc files include Everett, a program by Ted Dustman for initial processing of mapping data, Matmap, a set of MATLAB utilities by Jeroen Stinstra with a similar functionality, and tsdflib (as yet undocumented), a library created by Ted Dustman, Rob MacLeod, Jenny Simpsons, and Jeroen Stinstra that provides C-language access to container files.

For more information on container files, see the documentation for the graphicsio library at www.cvrti.utah.edu/~macleod/docs/graphicsio/.

6.3.3 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 that mimic the information available in the graphicsio .tsdf file so—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.

There will be more of these as we develop the format further so stay tuned.

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`

```matlab
>> sockdata.potvals = sockinfo(1:128,:);
>> sockdata.unit = 'mV'
>> save mysockdata.mat sockdata
```

6.3.4 Fiducial (ascii) files

A fiducial can be input currently in two ways: via a .tsdfc file, where the potential and fiducial values are stored together, or via a .fids file, a simple ASCII file containing the values for each node.

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 .geom 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) 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.
6.4 Channels and leadlinks

Indirections in map3d

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 “V1”.

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 an 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 .geom files Information for the channels array is stored as an associated scalar with the information in the standard .geom files. At present, there is no leadlinks array stored in the .geom file but this could change in the future.
6.4 Channels and leadlinks

**In .mat geometry files**  MATLAB geometry files can also contain an `channels` array stored as a field in the structure. See Section 6.1.4 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 9 in Section 8.2.3.

**In .channels files** A `channels` file is an ASCII file, the first record of which contains 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
```

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:

<table>
<thead>
<tr>
<th>Line number</th>
<th>Contents</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>nsegs</td>
<td>number of landmark segments in the file</td>
</tr>
<tr>
<td>2</td>
<td>1 type nsegpts seglabel</td>
<td>segment number (1), type, number of points, label_number</td>
</tr>
<tr>
<td>3</td>
<td>X, Y, Z, radius</td>
<td>point location and radius of point 1</td>
</tr>
<tr>
<td>4</td>
<td>X, Y, Z, radius</td>
<td>point location and radius of point 2</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>nsegs + 2</td>
<td>X, Y, Z, radius</td>
<td>point location and radius of last point in segment 1</td>
</tr>
<tr>
<td>.</td>
<td>2 type nsegpts seglabel</td>
<td>segment number (2), segment type, number of points, and label</td>
</tr>
<tr>
<td>.</td>
<td>X, Y, Z, radius</td>
<td>point location and radius of point 1</td>
</tr>
<tr>
<td>.</td>
<td>X, Y, Z, radius</td>
<td>point location and radius of point 2</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>X, Y, Z, radius</td>
<td>point location and radius of last point in segment 2</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

The landmark types defined to date are the following:
### 6.5 Landmark files

<table>
<thead>
<tr>
<th>Name</th>
<th>Graph. object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artery</td>
<td>faceted pipe</td>
<td>a coronary artery/vein segment</td>
</tr>
<tr>
<td>Occlusion</td>
<td>coloured sphere</td>
<td>an experimental occlusion that could be open and closed</td>
</tr>
<tr>
<td>Closure</td>
<td>coloured sphere</td>
<td>a permanent occlusion that cannot be opened</td>
</tr>
<tr>
<td>Stimulus</td>
<td>coloured sphere</td>
<td>a stimulus site</td>
</tr>
<tr>
<td>Lead</td>
<td>coloured sphere</td>
<td>a particular electrode or lead location</td>
</tr>
<tr>
<td>Plane</td>
<td>rectangular parallelepiped</td>
<td>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).</td>
</tr>
<tr>
<td>Rod</td>
<td>lines</td>
<td>rod inserted into needle track to digitize needle electrode locations.</td>
</tr>
<tr>
<td>PaceNeedle</td>
<td>sphere</td>
<td>location of a pacing needle entry point</td>
</tr>
<tr>
<td>Cath</td>
<td>faceted pipe</td>
<td>location of catheter in a vessel</td>
</tr>
<tr>
<td>Fiber</td>
<td>arrow</td>
<td>local fiber direction indicator</td>
</tr>
<tr>
<td>RecNeedle</td>
<td>sphere</td>
<td>location of recording needle entry point</td>
</tr>
<tr>
<td>Cannula</td>
<td>tube</td>
<td>coronary vessel cannulus</td>
</tr>
</tbody>
</table>

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 drawsurfmarks.c routine. At present, the values used are:

- Occlusion cyan
- Closure blue
- Stimulus yellow

and they are not (yet) adjustable by the user.

To specify a plane landmark requires three “points”

<table>
<thead>
<tr>
<th>Point</th>
<th>X,Y,Z</th>
<th>Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First point of plane</td>
<td>Radius of the plane</td>
</tr>
<tr>
<td>2</td>
<td>Second point of plane</td>
<td>Thickness of the plane</td>
</tr>
<tr>
<td>3</td>
<td>Third point of plane</td>
<td>not used</td>
</tr>
</tbody>
</table>

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 8.5.1). 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.

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. We 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 scaling scope uses the current user-selected values for maximum and minimum for the scaling (see -pl and -ph input parameters in Section 4). 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 ‘refsec:scalarparams for details.

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 = Kφ; the
7.4 Surface Data Display

The logarithmic model, which highlights the lower level data values at the cost of poorer resolution at the higher levels, i.e., \( \text{colour} = A \log(\phi) + B \); 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., \( \text{colour} = A e^{B\phi} \).

The two schemes with fixed numbers of contours, \( \log/7\text{-levels} \) and \( \log/13\text{-levels} \) both map the upper decade (\( \phi_{\text{max}} \) to \( \phi_{\text{max}}/10 \)) 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 \( \text{map3d} \). 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.

Contour spacing the contour values are a function of the data and the user selection of scale range, model, and mapping (see following items). Fundamentally, the user selects between contour spacing based on the number of contours selected or based on fixed spacing between contours. The actual result depends, in turn, on the range of data values and the desired mapping between value and colour.

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 \( \text{map3d} \) supports:

Mean as reference: Selecting the mean as reference causes \( \text{map3d} \) 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 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.

7.8 Time signal display

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

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 8.3.
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,

Scaling: scaling of value to colour and to contour line values,

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:

<table>
<thead>
<tr>
<th>Multi-surface layout</th>
<th>Selection method</th>
</tr>
</thead>
<tbody>
<tr>
<td>One surface per window</td>
<td>Mouse location establishes currently active window.</td>
</tr>
<tr>
<td>Several surfaces in one window</td>
<td>Up/down arrows selects the surface. Hitting the up-arrow key after selecting the last surface selects all surface.</td>
</tr>
</tbody>
</table>

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:
8.2 Mouse control, keyboard mapping, dials, and menus

**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 launching, 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 resized and moved 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 CTRL+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.](image)

**In surface windows:** when the mouse is over a surface window, mouse buttons have the following actions:
8.3 Controlling the time signal window

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

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

<table>
<thead>
<tr>
<th>Control Key</th>
<th>Button</th>
<th>Mouse Actions</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt(CTRL+SHIFT)</td>
<td>Left</td>
<td>Move a single surface subwindow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Middle</td>
<td>Resize single surface subwindow (no indication of change until release of mouse button).</td>
<td>no action</td>
</tr>
<tr>
<td></td>
<td>Right</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: if *map3d* does not respond as described in these tables, it could 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. To make such changes under IRIX, examine the .4Dwmrc file; 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. A list of the keyboard keys and their functions is shown in table 1 and table 2 describes the action for each of the function and arrow keys, and table 3 the actions of the keypad keys. The keys affect action only in the Geometry Window unless otherwise specified.

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.

Following are the controls for the menus of the Time Signal Window, Legend Window, and Main Window, respectively.

8.3 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 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.
8.3 Controlling the time signal window

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.

<table>
<thead>
<tr>
<th>Regular keyboard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a/A-key</td>
<td>Switch colour tables</td>
</tr>
<tr>
<td>c-key</td>
<td>Toggle contour draw</td>
</tr>
<tr>
<td>d-key</td>
<td>Toggle depth cueing</td>
</tr>
<tr>
<td>f-key</td>
<td>Toggle frame lock (also in Time signal window)</td>
</tr>
<tr>
<td>g-key</td>
<td>Toggle style of gouraud shading (between texture-mapped and non-texture-mapped)</td>
</tr>
<tr>
<td>i-key</td>
<td>Toggle “direction” (invert) of color table</td>
</tr>
<tr>
<td>l-key</td>
<td>Toggle use of lighting</td>
</tr>
<tr>
<td>m/M-key</td>
<td>Step through mesh/node drawing options</td>
</tr>
<tr>
<td>n-key</td>
<td>Toggle display of node labels (some node marking must be selected)</td>
</tr>
<tr>
<td>p-key</td>
<td>Toggle information in time signal (pick) window (also in Time Signal Window)</td>
</tr>
<tr>
<td>q-key</td>
<td>Quit or destroy a sub-window - Legend Window and Time Signal Window only (Escape quits the whole program)</td>
</tr>
<tr>
<td>r-key</td>
<td>Reset to startup conditions</td>
</tr>
<tr>
<td>R-key</td>
<td>Reset shading model (to wireframe rendering)</td>
</tr>
<tr>
<td>s/S-key</td>
<td>Cycle through the various surface data draw options</td>
</tr>
<tr>
<td>t-key</td>
<td>Toggle transformation lock</td>
</tr>
<tr>
<td>w-key</td>
<td>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 Save... Dialog (Section 8.5.2).</td>
</tr>
<tr>
<td>x-key</td>
<td>Draw axis</td>
</tr>
<tr>
<td>Escape</td>
<td>Quit the program, if pressed in a Geometry Window, or Destroys a Time Signal Window if pressed there.</td>
</tr>
<tr>
<td>+/- key</td>
<td>Increases/Decreases size of of currently-selected object (see Section 8.4.3)</td>
</tr>
<tr>
<td>(/) key</td>
<td>Changes which susrface 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)</td>
</tr>
</tbody>
</table>

Clipping Controls

<table>
<thead>
<tr>
<th>Ctrl</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -key</td>
<td>Toggle front clipping plane</td>
</tr>
<tr>
<td>&gt; -key</td>
<td>Toggle rear clipping plane</td>
</tr>
<tr>
<td>[ / ] -key</td>
<td>Move front clipping plane in (initially) +z/-z direction respectively</td>
</tr>
<tr>
<td>{ / } -key</td>
<td>Move rear clipping plane in (initially) +z/-z direction respectively</td>
</tr>
<tr>
<td>, -key</td>
<td>Lock/Unlock clipping plane rotation with object rotation (when unlocked, shift-Middle-click rotates clipping planes)</td>
</tr>
<tr>
<td>. -key</td>
<td>Lock/Unlock clipping planes from each other. When active, clipping planes move together</td>
</tr>
</tbody>
</table>

Table 2: Control of map3d via the arrow keys

<table>
<thead>
<tr>
<th>Arrow Keys</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left Arrow Key</td>
<td>Retreat by current frame step (Also in Time Signal Window)</td>
</tr>
<tr>
<td>Right Arrow Key</td>
<td>Advance by current frame step (Also in Time Signal Window)</td>
</tr>
<tr>
<td>Up Arrow key</td>
<td>Select next surface</td>
</tr>
<tr>
<td>Down Arrow Key</td>
<td>Select previous surface</td>
</tr>
</tbody>
</table>
### 8.3 Controlling the time signal window

<table>
<thead>
<tr>
<th>Keypad Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-Keypad Left-arrow</td>
</tr>
<tr>
<td>Ctrl-Keypad Right-arrow</td>
</tr>
<tr>
<td>Ctrl-Keypad Up-arrow</td>
</tr>
<tr>
<td>Ctrl-Keypad Down-arrow</td>
</tr>
<tr>
<td>Ctrl-Keypad Home</td>
</tr>
<tr>
<td>Ctrl-Keypad PgUp</td>
</tr>
<tr>
<td>Ctrl-Keypad End</td>
</tr>
<tr>
<td>Ctrl-Keypad PgDn</td>
</tr>
<tr>
<td>Alt(or CRTL+SHIFT)-Keypad Left-Arrow</td>
</tr>
<tr>
<td>Alt(or CRTL+SHIFT)-Keypad Right-Arrow</td>
</tr>
<tr>
<td>Alt(or CRTL+SHIFT)-Keypad Down-Arrow</td>
</tr>
<tr>
<td>Alt(or CRTL+SHIFT)-Keypad Up-Arrow</td>
</tr>
<tr>
<td>Alt(or CRTL+SHIFT)-Keypad Home</td>
</tr>
<tr>
<td>Alt(or CRTL+SHIFT)-Keypad PgUp</td>
</tr>
<tr>
<td>Plus/Minus key</td>
</tr>
</tbody>
</table>

Table 3: 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.

<table>
<thead>
<tr>
<th>Overview of Geometry Window menus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
</tr>
<tr>
<td>Save</td>
</tr>
<tr>
<td>Contours</td>
</tr>
<tr>
<td>Fiducials</td>
</tr>
<tr>
<td>Frame Controls</td>
</tr>
<tr>
<td>Graphics</td>
</tr>
<tr>
<td>Landmarks</td>
</tr>
<tr>
<td>Mesh</td>
</tr>
<tr>
<td>Node marking</td>
</tr>
<tr>
<td>Picking</td>
</tr>
<tr>
<td>Scaling</td>
</tr>
<tr>
<td>Surface Data</td>
</tr>
<tr>
<td>Use +/- to select</td>
</tr>
<tr>
<td>Window Attributes</td>
</tr>
</tbody>
</table>

Table 4: The overall menu structure of the Geometry 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 than beginning at 0 or 1 and going to 10 (the number of frames of data actually read).
8.3 Controlling the time signal window

### Contour Menus

<table>
<thead>
<tr>
<th>Number of Contours</th>
<th>User-specified spacing</th>
<th>use the value of contour spacing from the (-cs) option of the command line, or value in the Contour Spacing dialog.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draw style</td>
<td>Dashed line for negative values</td>
<td>draw positive contours in solid, negative in broken lines</td>
</tr>
<tr>
<td>Set Contour Spacing</td>
<td>Solid lines for all contours</td>
<td>draw all contours in solid lines</td>
</tr>
<tr>
<td>Line size</td>
<td>set the line thickness</td>
<td></td>
</tr>
<tr>
<td>Toggle contours</td>
<td>toggle display of contours without changing settings</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Menus for contour spacing/number.

### Frame Control Menus

<table>
<thead>
<tr>
<th>Lock Frames</th>
<th>toggle whether frames operations affect one surface or all surfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Frame Interval</td>
<td>Set Frame Step</td>
</tr>
<tr>
<td></td>
<td>User-specified Interval value</td>
</tr>
<tr>
<td></td>
<td>opens a dialog to interactively set User-specified frame step. Affected surfaces are determined by the lock status.</td>
</tr>
<tr>
<td></td>
<td>use the value of interval specified in the (-i) option of the command line or the value set in the frame step dialog.</td>
</tr>
<tr>
<td></td>
<td>select between 1 and 90 for frame animation step</td>
</tr>
<tr>
<td></td>
<td>positions the surface at the first position in time</td>
</tr>
<tr>
<td></td>
<td>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)</td>
</tr>
<tr>
<td></td>
<td>Set current frame to be time zero.</td>
</tr>
</tbody>
</table>

Table 6: Frame Controls.

#### 8.3.1 Adjusting the frame marker

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, \textit{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.
8.4 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. While the Color and Size Picker leave much to be desired aesthetically, they improve on past versions, and have slightly more...
### Node Marking Menus

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

Table 9: 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 lock section (see Section 8.4.3). If we have a single or current) surface only, then change only that surface.

8.4.1 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...
8.4 Color/Size Selection

## Picking Menus I

<table>
<thead>
<tr>
<th>Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Signal (new window mode)</td>
<td>Create a new time signal window with each pick of a node</td>
</tr>
<tr>
<td>Time Signal (refresh window mode)</td>
<td>Update the last time signal window with each pick of a node</td>
</tr>
<tr>
<td>Display node info mode</td>
<td>Picking will cause certain node information to be dumped to the console.</td>
</tr>
<tr>
<td>Display triangle info mode</td>
<td>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.</td>
</tr>
<tr>
<td>Triangle construction/deletion</td>
<td>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.</td>
</tr>
<tr>
<td>Flip triangle mode</td>
<td>Will change the order of drawing the triangle’s points. This will cause front-facing triangles to become back-facing and vice-versa.</td>
</tr>
<tr>
<td>Edit node mode</td>
<td>Will allow you to pick a point and translate it with the keyboard transform controls (see Section 8.2.2, keypad controls)</td>
</tr>
<tr>
<td>Edit landmark point mode</td>
<td>Will allow you to pick a landmark point and translate it with the keyboard transform controls (see Section 8.2.2, keypad controls)</td>
</tr>
<tr>
<td>Delete node mode</td>
<td>Will remove from the geometry any node that you pick and any triangles associated with it.</td>
</tr>
</tbody>
</table>

Table 10: 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.

Figure 4: Color Picker.

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.

### 8.4.2 Size Picker

Figure 5: 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
Table 11: 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.

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.

### 8.4.3 Interactive Size Control

Rather than having to open the Size Picker over and over, \textit{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 Interactive GUIs - File Selection, File Saving, and Scaling Options, etc,

This version of \textit{map3d} adds new GUI support. The most noticeable new addition is the “\textit{map3d} Files” window, allows you to interactively select filenames, data windows, etc. The others are for saving files, and for quick scaling changes.

#### 8.5.1 Files Window

The “\textit{map3d} Files” window can be accessed by 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 you make will take effect until you click on the “Apply” button. If you click on the “Close” button, the window will close, but if you open it again, it will look exactly as when you closed it.

Also note that this window allows you to run \textit{map3d} without any arguments. If you do so, or if \textit{map3d}
### Scaling Menus

<table>
<thead>
<tr>
<th>Scaling...</th>
<th>Range</th>
<th>Function</th>
<th>Mapping</th>
<th>Grouping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td></td>
<td>Linear</td>
<td>True</td>
<td>Move to group #</td>
</tr>
<tr>
<td>Global over all frames in one surface</td>
<td></td>
<td>Linear</td>
<td>Symmetric about zero</td>
<td>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)</td>
</tr>
<tr>
<td>Global over all surfaces in one frame</td>
<td></td>
<td>Logarithmic</td>
<td>Symmetric about zero</td>
<td></td>
</tr>
<tr>
<td>Global over all surfaces and frames</td>
<td></td>
<td>Exponential</td>
<td>Symmetric about zero</td>
<td></td>
</tr>
<tr>
<td>Scaling over groups in one frame</td>
<td></td>
<td>Lab standard</td>
<td>Symmetric about zero</td>
<td></td>
</tr>
<tr>
<td>Scaling over groups in all frames</td>
<td></td>
<td>Lab 13 standard</td>
<td>Symmetric about zero</td>
<td></td>
</tr>
<tr>
<td>Slave Scaling over one frame</td>
<td></td>
<td>True</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>Slave Scaling over all frames</td>
<td></td>
<td>Symmetric about zero</td>
<td>Symmetric about zero</td>
<td></td>
</tr>
<tr>
<td>Command-line specified range</td>
<td></td>
<td>Symmetric about zero</td>
<td>Symmetric about zero</td>
<td></td>
</tr>
</tbody>
</table>

Table 12: Menu for scaling, the mapping from data value to color for rendering.

Figure 6: Files Dialog for *map3d*.
8.5 Interactive GUIs - File Selection, File Saving, and Scaling Options, etc, CVRTI/SCI Manual: *map3d*

<table>
<thead>
<tr>
<th><strong>Surface Display Menus</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Color</strong></td>
</tr>
<tr>
<td>Rainbow</td>
</tr>
<tr>
<td>Green to red</td>
</tr>
<tr>
<td>White to Black</td>
</tr>
<tr>
<td>Invert</td>
</tr>
</tbody>
</table>

- **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.
- **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 13: Submenus to control the display of scalar data on the mesh.

<table>
<thead>
<tr>
<th><strong>+/- Adjust Menu</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Large Font Size</strong></td>
</tr>
<tr>
<td><strong>Medium Font Size</strong></td>
</tr>
<tr>
<td><strong>Small Font Size</strong></td>
</tr>
<tr>
<td><strong>Contour Size</strong></td>
</tr>
<tr>
<td><strong>Line/Point Size</strong></td>
</tr>
<tr>
<td><strong>Node Marks (all) Size</strong></td>
</tr>
<tr>
<td><strong>Node Marks (extrema) Size</strong></td>
</tr>
<tr>
<td><strong>Node Marks (time signal) Size</strong></td>
</tr>
<tr>
<td><strong>Node Marks (leads) Size</strong></td>
</tr>
<tr>
<td><strong>Change in translation</strong></td>
</tr>
<tr>
<td><strong>Change in scaling</strong></td>
</tr>
<tr>
<td><strong>Change in rotation</strong></td>
</tr>
</tbody>
</table>

Table 14: The Use +/- to interactively change sizes menu

Determines that none of the geometry files were valid, the files window will pop up with an empty row waiting for input. If you try to close this window without any input, *map3d* will treat is as though you were exiting the program.

![Figure 7: Empty Files Dialog. *map3d* will start up this way if you run without arguments.](image)
### Window Attributes Menus

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

Table 15: Controls for the attributes of the `map3d` windows.

### Time Signal Window Controls

<table>
<thead>
<tr>
<th>Axes Color</th>
<th>Select Axes Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Color</td>
<td>Color of data graph</td>
</tr>
<tr>
<td>Toggle Display Mode</td>
<td>Show basic values and graph, show detailed values and graph, or show larger graph</td>
</tr>
</tbody>
</table>

Table 16: Controls for the attributes of the Time Signal Window.

### Legend Window Controls

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

Table 17: Controls for the attributes of the Legend window.
8.5 Interactive GUIs - File Selection, File Saving, and Scaling Options, etc, CVRTI/SCI Manual: map3d

Main Window Controls

| Set Background Color | Select bg Color |
| Quit Map3D           | Quit Map3D      |

Table 18: Controls for the Legend Window Menu

File Window Options

<table>
<thead>
<tr>
<th>Surf</th>
<th>Number of the surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win#</td>
<td>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).</td>
</tr>
<tr>
<td>Geom File</td>
<td>Name of the geometry file of this surface. You may change the geom file by clicking on the ... button and finding the file you want in the file picking window that will appear. You may reload the current geometry by selecting the “Mesh-¿Reload Geometry” menu entry.</td>
</tr>
<tr>
<td>Geom#</td>
<td>Number of surface within the geometry file (see Section 6.1.3). If there is more than one surface in the geometry file, you may select which one you want from the drop-down menu. If you change the geometry file, this menu will be updated.</td>
</tr>
<tr>
<td>Time Series File</td>
<td>Name of the data file of this surface. You may change the data file by clicking on the ... button and finding the file you want in the file picking window that will appear. You may reload the current geometry by selecting the “Mesh-¿Reload Data” menu entry.</td>
</tr>
<tr>
<td>Start Frame</td>
<td>Start reading data at this frame. You can also left-click in the graph section and drag to select the correct frame and this number will be updated.</td>
</tr>
<tr>
<td>End Frame</td>
<td>Finish reading data at this frame. You can also middle-click in the graph section and drag to select the correct frame and this number will be updated.</td>
</tr>
<tr>
<td>Graph</td>
<td>Graph of the root-mean-square deviation of each node for each time instance. You can left- or middle-click in this graph to select your data window, (and the number in start frame or end frame will update).</td>
</tr>
<tr>
<td>Other Files</td>
<td>This button will pop open another window in which you can view/modify the channels, leadlinks, landmarks, or fiducial file to use with this surface.</td>
</tr>
</tbody>
</table>

Table 19: Options for the file window

8.5.2 Saving Files

The “Save...” Window will display if you select “Window Attributes-¿Save...” from the menu. The first section is to save geometry files, the second is for images, and the third for settings.

Saving Geometry In the Geometry section it displays each surface number with its current geometry filename. Next to each there is a check box. If that check box is checked when you click one of the save buttons, then that surface will be one of the ones that is saved. If the second check box (labelled Transform?) is checked, then map3d will save the geometry will be saved with the current transformations.
8.5 Interactive GUIs - File Selection, File Saving, and Scaling Options, etc, CVRTI/SCI Manual: *map3d*

Figure 8: Saving Dialog.

(translations and rotations) applied.

Make sure to modify the filename or the current filename will be replaced (if you click on ... you can browse for a filename). If the filename ends in the extension .fac, it will save filename.fac and filename.pts. If the filename ends in .geom, it will save the file in the CVRTI binary file format (see Section 6.1.3). If Transforms is selected and there is a landmarks file, then a filename.lmark.

Clicking on the “Save in 1 geom file” button will save all of the geometry into 1 CVRTI binary file (see Section 6.1.3), choosing the filename of the first entry, which must have a .geom extension. Clicking “Save individually” will save each in its own filename. There is currently a known bug with saving .geom files under Windows and reading them back in again.

**Saving Images** In the image section, 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 save is approximately the space that all open map3d windows cover.

NOTE: If you have windows that overlap, the one on top might not be the one *map3d* thinks is on top. So if you move windows around, click inside the window (not the title bar), to tell *map3d* it is on the top. Otherwise, data you want might be overridden by the data in the window underneath.

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** This section 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 four controls to select when 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.

4. Save Frame Every x milliseconds - A frame will be saved approximately every x milliseconds, depending on whether map3d is doing something else (like saving an image for one of the other controls). This option will only take effect if you select it while animations are not being saved.

Naturally, animations start recording when you click the “Start saving animations” button, and end when you click the “Stop saving animations” button. If you close the window while animations are being saved, they will keep recording (this is useful if the Save... window gets in the way of the images you want to save).

**Saving Settings** map3d saves two types of settings files. There is the .map3drc file, which acts as a settings file which (if in your home directory or in the directory in which you run map3d) will load a list of options at start-time, so map3d can behave similarly each time you run it without having to reset options manually.

The other type of setting file is a script (or batch) file (See Section sec:scripts). map3d will attempt to save exactly what you have loaded, including all files and window positions. The difference between a script and a batch file is that a batch file is designed for MS Windows, and a script is designed for everything else.

To save a script/batch file, click on the appropriate button and select a filename. If you choose to save a .map3drc file, you cannot change the filename.

The only real difference between scripts and the .map3drc file is that the .map3drc file saves global options and the options set on the first surface (which then will apply to all surfaces), and scripts save information about all surfaces. The options they save are the same, so they are grouped together.

These options are added to the command line. For the script, the options are saved as command line arguments, and for the .map3drc file the options are saved to a file and when map3d is executed, the options get inserted into the command line before any other arguments. Note that this also happens when you run map3d from a script. Also note that the options set in the .map3drc file may be overridden by subsequent arguments. (Naturally, both the .map3drc file and the script file may be edited by hand.)

These are three types of options that map3d can save: global options, which are set independent of any surface; surface transformations, which will save the rotation, translation, and scaling of the surfaces; and other surface options, which save everything else. You may select or deselect any of these three items as you wish. Note, if none of them is selected and save a .map3drc file, it will be empty. If none is selected and you save a script, then it will save the filenames and the window information.

The options are as follows (some of these are mentioned in the usage - Section 4). Some of the values are unclear (like the -sc scaling option) as to what the result will be if you edit it by hand. Future versions of map3d will make this more clear. See Section 7 for more information on these features.

### 8.5.3 Scaling Options

This dialog is pretty much a duplicate of the Scaling Menu in the Geometry menu, but in dialog form for convenience sake. 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)

### 8.5.4 Contour Spacing Dialog

This dialog allows you to select the contour spacing, number of contours and scaling range. A side-effect of this dialog is that if you change one feature, it will change another feature. I.e., if you change the number of contours, then it will also change the spacing.
### Global Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-b</code></td>
<td>Run <code>map3d</code> in borderless mode</td>
</tr>
<tr>
<td><code>-nw</code></td>
<td>Assign each surface its own window</td>
</tr>
<tr>
<td><code>-sc range function mapping</code></td>
<td>Sets the global range, function, and mapping based on each number. Range can be a number from 0-8, function from 0-4, mapping from 0-2.</td>
</tr>
<tr>
<td><code>-l general transform frame</code></td>
<td>General, transform, and frame are 0 or 1 based on whether or not the general, transform, and frame locks are set, respectively.</td>
</tr>
<tr>
<td><code>-rl level</code></td>
<td>Sets the report level to level</td>
</tr>
<tr>
<td><code>-pm mode</code></td>
<td>Sets the pick mode to mode (mode is a number from 0-12).</td>
</tr>
</tbody>
</table>

### Surface Transformation Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-rq w x y z</code></td>
<td>Rotation Quaternion. These are 4 floating-point values which represent the rotation info</td>
</tr>
<tr>
<td><code>-tc x y z</code></td>
<td>X, y, z translation coordinates</td>
</tr>
<tr>
<td><code>-zf factor</code></td>
<td>Zoom factor. This really applies to the window as opposed to the surface</td>
</tr>
</tbody>
</table>

### Other Surface Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-c r g b</code></td>
<td>Mesh Color, red, green, blue values</td>
</tr>
<tr>
<td><code>-sm mode</code></td>
<td>Shading model (flat, gouraud, banded) - mode is a number between 0-3</td>
</tr>
<tr>
<td><code>-rm mode</code></td>
<td>Mesh render mode (dots, connectivity, etc.) - mode is a number between 0-5</td>
</tr>
<tr>
<td><code>-ic mode</code></td>
<td>Inverted Colormap - 1 for true, 0 for false</td>
</tr>
<tr>
<td><code>-slw mode</code></td>
<td>Show colormap legend window - 1 for true, 0 for false</td>
</tr>
<tr>
<td><code>-el mode</code></td>
<td>Lighting enabled - 1 for true, 0 for false</td>
</tr>
<tr>
<td><code>-ef mode</code></td>
<td>Fogging (depth cue) - 1 for true, 0 for false</td>
</tr>
<tr>
<td><code>-sco mode</code></td>
<td>Contours shown - 1 for true, 0 for false</td>
</tr>
<tr>
<td><code>-nc mode</code></td>
<td>Negative contours dashed - 1 for true, 0 for false</td>
</tr>
<tr>
<td><code>-x mode</code></td>
<td>Axes drawn - 1 for true, 0 for false</td>
</tr>
<tr>
<td><code>-xc r g b</code></td>
<td>Axes Color, red, green, blue values</td>
</tr>
</tbody>
</table>

Table 20: Settings options
As a result, there are three buttons on the top of the dialog to select which feature you want to hold constant. So if you select "Keep spacing constant", and change the number of contours, then it will compute a new range based on the new number of contours and spacing; if you select "Keep num contours constant" and change the range, it will compute a new contour spacing.

Note: for the resulting range to be in effect, the “Scaling range” (see Section:sec:scalinggui) must be set to “user-defined”.

8.5.5 Fiducial Control

This dialog allows the user to select which Fiducial Contour and/or Fiducial Contour Map he wants to visualize. Under the Fiducial Contour tab the user can select to visualize the Activation Time contour and/or Recovery Time contour or No Fiducials. Under the Fiducial Map tab the user can select to visualize the Activation Time contour map or Recovery Time contour map or No Fiducial maps. The user can also change the contour spacing of the selected contour map.

8.5.6 Other dialogs

8.5.7 Other dialogs

We are getting many interactive windows in map3d. There are too many to document for each individual feature they control and to show a picture, so instead we will document them in the menu section that
8.6 Picking mode

By “picking” we mean selecting some piece of the display in the current window using the mouse (with buttons). *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. In version 5.4, the choices are limited to node information, triangle information, time signal displays, triangulation, triangle deletion, and triangle flipping. 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 8.5.2). To control picking, use the top-level “Picking” menu. See Tables 10 and 11 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
8.7 Control of landmark display

There are some lighting and colour controls for the display of landmarks that are useful to know about. Table 21 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.
<table>
<thead>
<tr>
<th>Landmarks Menus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Coronary/Catheter</strong></td>
</tr>
<tr>
<td>Toggle Coronary</td>
</tr>
<tr>
<td>Toggle Catheter</td>
</tr>
<tr>
<td>Wireframe Coronary</td>
</tr>
<tr>
<td>Coronary Color</td>
</tr>
<tr>
<td>Catheter Color</td>
</tr>
<tr>
<td><strong>Points</strong></td>
</tr>
<tr>
<td>Toggle temporary occlusions</td>
</tr>
<tr>
<td>Toggle permanent occlusions (stich)</td>
</tr>
<tr>
<td>Toggle stimulation site (lead)</td>
</tr>
<tr>
<td>Toggle recording site (lead)</td>
</tr>
<tr>
<td>Occlus Color</td>
</tr>
<tr>
<td>Stitch Color</td>
</tr>
<tr>
<td>Stim Color</td>
</tr>
<tr>
<td>Lead Color</td>
</tr>
<tr>
<td><strong>Planes</strong></td>
</tr>
<tr>
<td>Toggle Plane</td>
</tr>
<tr>
<td>Toggle Plane Transparency</td>
</tr>
<tr>
<td>Plane Color</td>
</tr>
<tr>
<td><strong>Points</strong></td>
</tr>
<tr>
<td>Toggle rod</td>
</tr>
<tr>
<td>Toggle recording needle</td>
</tr>
<tr>
<td>Toggle pace needle (lead)</td>
</tr>
<tr>
<td>Toggle fiber (lead)</td>
</tr>
<tr>
<td>Toggle cannula (lead)</td>
</tr>
<tr>
<td>Rod Color</td>
</tr>
<tr>
<td>Recneedle Color</td>
</tr>
<tr>
<td>Paceneedle Color</td>
</tr>
<tr>
<td>Fiber Color</td>
</tr>
<tr>
<td>Cannula Color</td>
</tr>
<tr>
<td><strong>Toggle All Landmarks</strong></td>
</tr>
<tr>
<td><strong>Points</strong></td>
</tr>
<tr>
<td>Toggle All Points</td>
</tr>
<tr>
<td><strong>Toggle All Rods</strong></td>
</tr>
</tbody>
</table>

Table 21: Landmark menus. These control landmark display options.

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
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 8.5.2). 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 8.5.2 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 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, which comes free on a Mac. It is worth upgrading to the iLife version 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. mediaconvert (for SGI), see man mediaconvert.
2. QuickTimePro (for Mac OSX), but if you want this one, you’ll have to pay.
3. Discreet Cleaner XL (for Win32), you’ll also have to pay for this one.
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.
10 BUGS

Too many to even begin to contemplate. But if there are any you would like exterminated, please send email to map3d@sci.utah.edu (we accept all foreign currency in large denominations, bicycle parts, assorted outdoor gear, but no credit cards).

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

- There is some trouble in opening .geom files saved from the Windows version of map3d.
References


