map3d User's Guide

Version 5.2

Last update: July 23, 2001

Authors: Rob MacLeod (macleod@cvrti.utah.edu),
Chris Moulding (moulding@cs.utah.edu)
and
Bryan Worthen (worthen@cs.utah.edu)

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

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

Center for Bioelectric Field Modeling, Simulation and Visualization
www.sci.utah.edu/ncrr

the University of Utah

Support for this project came from:

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

Contents

1 What's New?

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

1.1 Version 5.2: July 2001

This is the first major revision of the "new" *map3d* and we are getting close to the functionality of the old GL based version. There are also some features unique to the OpenGL version like the lighting model and excellent mouse controls of rotation.

Some of the specific things that you should notice over the version 5.1 include:

Time signal window: we have redesigned the time signal window and it now permits selection of the time instant displayed in the surface windows. This is important functionality for the use of *map3d* for time sequences. We have also changed the fonts and the layout of the window to make more room for the signal. With the "p" key, one can even turn off all the displayed information in the time signal window and see only the data. Section 8.3

Frame control: control the alignment of time signals between surface windows, adjust zero, alter step between time frames. Section 8.3.1

Clipping planes: there are two clipping planes available and they can even be locked to provide slicing functions. Section 7.6

Locks: there are now three lock icons on the surface display and they tell the user a) (yellow) whether display parameters affect all surfaces or just the current one, b) (red) whether rotations and translations apply to one or all surfaces, and c) (green) whether the frame counters in the time signals apply to all surfaces or just the current surface. Section 8.1.1

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:

- Landmarks
- Support for dial boxes
- Improvements in borderless windows
- Clean up the scaling windows
- Better looking fonts on everything
- Addition of a GUI to allow data windowing and better interactive selection of parameters
- New display modes for the display of vectors,

2 Introduction

This document describes the function and usage of the program *map3d*, a scientific visualization application developed at the Nora Eccles Harrison Cardiovascular Research and Training (CVRTI) and 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, 2, 3, 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.

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, Salt Lake City. This was my introduction to GL and C and this program became my personal sand box to play in. Along the way, Phil Ershler made valuable contributions in figuring out the magic of Formslib for some user interface controls and developing with me *graphicsio*, the geometry and data file library that supports *map3d*.

This is a new"new" version of *map3d*, the first to use the OpenGL library and thus become truly portable. It marks a new beginning for the program and the result of a collaborative effort from a new cast of contributors. The largest credit must go to Chris Moulding, an ace graphics programmer and general software whiz who surveyed my sand box architecture, pulled together the essential walls, created new ways to make rooms, and still left lots of the sand box around so I could continue to play. Guo Tang has recently added his prodigious programming skills to the project and is working on a few of his own rooms. From version 5.2 onward, Bryan Worthen deserves special notice as he has provided the energy and time to continue developing *map3d*.

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. 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.

We gratefully acknowledge the financial support that has come from the NIH, National Center for Research Resources (NCRR) the Nora Eccles Treadwell Foundation, and the University of Utah, which provides us with space and materials to create this sand box.

Rob MacLeod, July 23, 2001

3 Installation

3.1 System requirements

map3d is written in standard C/C++ and uses the OpenGL library, both choices made to ensure broad portability of the program. This first OpenGL version of *map3d* targets three platforms and we list the requirements and recommendations for each below. Additional platforms will follow—and follow even faster if you send us a machine to play on.

All platforms:

Requirements for all systems	
OpenGL libraries (GL and GLU)	version 1.1 + [1]
OpenGL/window interface library (GLX)	
Glut	version 3.7 +

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@cs.utah.edu)

Requirements	
Operating System	Irix 6.5+
Architecture	mips3 or mips4 (maybe mips2)
Applications Binary Interface	n32 or 64
Recommendations	
Hardware	Texture mapping hardware 128 MB main memory

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.

Requirements	
Operating System	kernel 2.2.x
Architecture	i386 (+ maybe PPC)
Applications Binary Interface	libc2.1
Recommendations	
Window system	XFree86 version 4.0 +
Hardware	3D graphics card (nVidia, 3dfx, ati) 128 MB main memory

LinuxPPC (PowerPC): *map3d* requires the OpenGL library, which is available as the mesa library at www.mesa3d.org for any Linux platform.

Requirements	
Operating System	kernel 2.2.x
Architecture	PowerPC (PPC)
Applications Binary Interface	libc
Recommendations	
Window system	XFree86 version 3.3 +
Hardware	3D graphics card) 128 MB main memory

Windows:

Requirements	
Operating System	W2K/NT4.0/9x
Architecture	i386
Applications Binary Interface	win32
Recommendations	
Hardware	3D graphics card (nVidia, 3dfx, ati) 128 MB main memory

3.2 Installation

The process of installing *map3d* is quite simple and we provide the steps in README files that comes with the downloaded installation files.

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*.

To download the software, use the URL

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

3.3 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 web site

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

3.4 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@cs.utah.edu or point your browser at `software.sci.utah.edu/research/software/bugzilla/` 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.

3.5 How to reach us

We have established an email address for *map3d*, map3d@cs.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@cs.utah.edu`. To subscribe to this list, send email to `majordomo@cs.utah.edu` with the content

```
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

```
map3d  -b -nw
       -f geomfilename
         -as 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
```

4.1 Typical usage examples

Here are some typical examples of using *map3d*:

- Display the contents of a geometry file:

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

  The first form 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
  ```

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

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

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

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

  ```
  map3d -f geomfilename.geom -p 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,

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

  ```
  map3d -f geomfilename.fac -p 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, but means of a suffix to the geometry filename specification. The calling string

  ```
  map3d -f geomfilename.geom@2 -p 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 `-nw` option creates a separate window for each of the surfaces. So a typical call would look like

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

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

This version of *map3d* does not provide an interactive means of specifying geometry numbers from a .geom file or time instants from a time series data file. If this information is not in the command line, the program takes the first surface and the entire time series, respectively. We will have this fixed soon, real soon. . .

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 individual windows, hold the Alt (meta) key and use the left and middle mouse buttons, respectively.

`-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.

4.3 Geometry specifications

The basis for display in *map3d* are 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 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* read the first surface in the file.

Note: by specifying a root filename without any extension, *map3d* will look for all valid geometry files and try and construct the most comprehensive set. Where there are multiple, potentially conflicting files with the same root, *e.g.,* `file.pts` and `file.geom`, *map3d* will ask the user for a decision of which to use. See Section6.2 for more details on the rules for specifying and reading geometry files. Note also that a geometry specification is the minimal input for successful function of *map3d*.

`-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.

`-c colour` = desired colour for the mesh of a particular surface, specified as a red, green, and blue value triplet ranging from 0 to 255 Check these values. Some examples are:

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

4.4 Scalar Data parameters

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

`-p potfilename` = base filename for the potential and current data files. For pot files (see Section 6.3.1 for details of the format), if the `-s` option is used, *map3d* will append a number and the extensions `.pot` to this base filename (see `-s` option). For binary time series (.tsdf) files, the `-s` option specifies the start and end frame numbers to be read from the file. With no `-s` option, *map3d* searches for a single .pot file named `potfilename.pot` or for a time series file named `potfilename.tsdf` and will read in all time instants from the file.

`-s num1 num2` = range of frame numbers to read. If we are reading data from .pot or .grad files, *map3d* appends each of the numbers between `num1` and `num2` to the value of `potfilename` to make complete pot filenames.
eg., `-p good-map -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.

`-i increment` = difference between each frame number. For .pot files this affects the string appended to potfilename for each new file.

eg., `-p good-map -s 1 6 -i 2` expands to:
`good-map001.pot good-map003.pot good-map005.pot`
For time series (.tsdf) data files, `increment` determines the step between frames actually ready from the file. Note that such a subsampling determines the data available in *map3d*, which no subsequent user action can alter.

`-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.

`-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.

`-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 the node number in the geometry. 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 8.4 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.

5 Script Files

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

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

5.1 What are script files?

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

5.2 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.,*

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

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

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

```
chmod 755 script_filename
```

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

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

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

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

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

```
geomdir=/u/macleod/torso/geom
geomfile=datorso.fac
datafile=dipole.data
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 concatneate text and variables you simply write them together (*e.g., geomdir*/`geomfile.pts` comcatenates 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.data \
               -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.data@1 -s 1 1000 \
               -ch ${GEOM}/sock128.channels \
               -f ${GEOM}/25feb97_sock_closed.geom \
               -p ${DATA}/cool1-atdr_new.data@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.data@1 -s 1 476 \
               -at 200 600 200 420 -t 9\
               -ch ${GEOM}/sock128.channels \
               -f ${GEOM}/25feb97_sock_closed.geom \
               -as 590 990 400 800 \
               -p ${DATA}/cool1-atdr_new.data@2 -s 1 476 \
               -at 590 990 200 420 -t 126 \
               -ch ${GEOM}/sock128.channels
```

`map3d-torso1.sh`**, included with this distribution**

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

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

`map3d-torso2.sh`**, included with this distribution**

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

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

Set some environment variables, then layout the whole display

```
#!/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.data@1 \
        -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.data@1 \
        -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.data@1 \
        -s 65 380 \
        -f $basedir/geom/525sock.geom \
        -as  476 800 176 500 \
        -at  476 800 50 175 -t 237 \
        -p $basedir/data/pace-center.data@1 \
        -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 may occur via a family of simple ASCII files used to describe geometry. These include 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. 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 the *graphicsio* library. Extensive documentation of this format is available from Rob MacLeod (macleod@cvrti.utah.edu) (www.cvrti.utah.edu/~macleod/docs) and from the CVRTI web site under the heading *graphicsio library* at www.cvrti.utah.edu/computing.shtml.

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.

map3d is capable of reading surface geometry from the *graphicsio* .geom files, at present only from single surfaces specified in the command line. Soon we will add support for reading more than one surface from a multi-surface .geom file.

6.2 Command line control of geometry files

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

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

A further way to read geometry into *map3d* is to use the geometry filename that can be optionally contained within the time series file (see Section 6.3) that contains the potentials. This requires that the .tsdf file be created with the geometry filename included; adding this after the fact is difficult. Note that even if a geometry filename is found in the .tsdf file, it can be overridden by the geometry file name specified in the argument list of *map3d*.

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) format files

The more 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. These files (.tsdf files) are capable of holding an entire sequence, or *time series* of scalar data in a single file, along with information about the contents, type, units, and temporal fiducials from the time series. For more details on this file format see www.cvrti.utah.edu/˜macleod/docs or the information under the *graphicsio library* at www.cvrti.utah.edu/computing.shtml.

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

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: The *graphicsio* library also defines a container file format that can describe a set of time series data (.tsdf) files, and can contain parameters extracted from the associated time series. At present, *map3d* does not read such container files, but this will change soon…

6.4 Channels and leadlinks

6.4.1 Description of leadlinks and channels information

Channels and leadlinks files [2], 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

[2]Note that at present, *map3d* does not support leadlinks files. We include them here because they belong to the full story and will appear in the next release

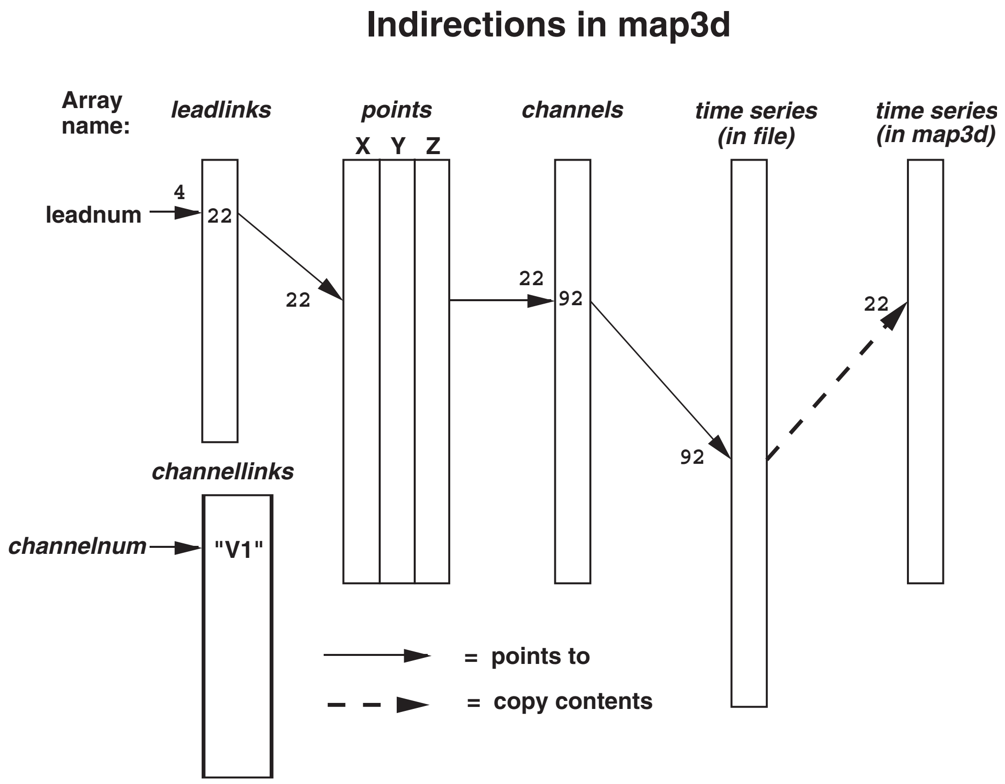

Figure 1: Example of the indirection possible in *map3d* through the use of *leadlinks channels*, and *channellinks*. Lead number 4 points, via the *leadlinks* array to node number 22. This, in turn, points via the *channels* array to location 92 in the multiplexed data buffer, which causes the values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

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.

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

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

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

6.4.2 Source of leadlink and channel information

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

In .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.

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 8 in Section 8.2.3.

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

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

For example, the file for a surface which reads:

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

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

6.4.3 Display of lead/channel information

To see how *map3d* can display the node and lead information, see Sections 7.7 and 8.

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 included into a single invocation of the program. At present, *map3d* will handle either multiple surfaces in a single window, or multiple windows with a single surface in each, but not multiple windows with more than one surface in each.

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 lay out and edit the shape and location of each of these sub-windows using the Alt(Meta) key and the left and middle mouse buttons. 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.

7.3 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.

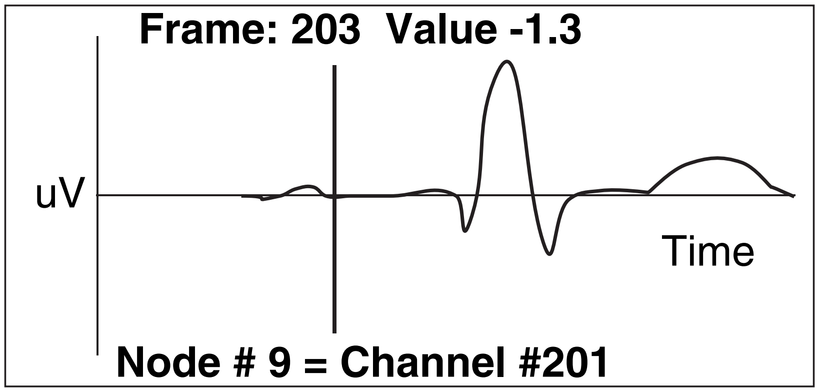

Figure 2: Time signal window layout. Vertical line indicates the frame currently displayed in the surface plot. Text annotations can vary with the data content and mode settings.

For directions on how to control the time signal window, see Section 8.3.

7.4 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: map 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.

A significant addition to this version of *map3d* is the ability to render all elements with a lighting model. This is especially useful for displaying the elements of the mesh. Addition controls to note are depth cueing, which can reveal the depth relationships between elements of the mesh.

7.5 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

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

Contours: these 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.

7.5.1 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). Not all these options are available in this version of *map3d*—see Section 8.2.3 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.,* $colour = K\phi$; the **logarithmic** model, which highlights the lower level data values at the cost of poorer resolution at the higher levels *i.e.,* $colour = A\log(\phi) + B$; and the **exponential** model, which does the opposite, compressing the smaller levels and expanding the higher ones to span a wider colour range, *i.e.,* $colour = Ae^{B\phi}$.

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

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

Rainbow map Colours range in rainbow fashion from dark 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.

Red (+) *and* Green (-) Red is once again positive and green negative, but there is no gradation with value, *i.e.,* this is a colour map of only two colours.

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.

7.6 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.

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:

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

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

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

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

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

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

8.2 Mouse control, keyboard mapping, dials, and menus

Direct interactive control of *map3d* is by the keyboard, mouse, and dials. 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 with the dials, if they are present, through dialogues, or by repeating keystrokes. Below are tables of all the current control devices and their function. When the program launches, the user sets one or more windows which can be resized and moved at any time. When launching the program with the `-b` option, the resulting borderless sub-window(s) can still be moved within a main window using the Alt-key together with the left and middle mouse buttons.

8.2.1 Mouse control

The mouse can be used for different purposes depending on the current mode. This is especially important when picking is selected. To shift between modes requires simultaneous pressing of the various modifier keys, shift, control, alt. Figure 3 shows the various actions of the mouse buttons.

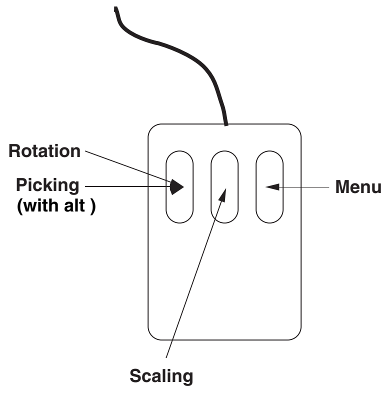

Figure 3: Mouse action for *map3d*. Mouse action of both the left and middle mouse buttons depends on the current mode. Picking makes intensive use of the mouse, as does moving objects in the surface window.

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

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

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

Mouse Actions		
Control Key	Button	Action
Alt	Left	Move a single surface subwindow (no indication of motion; release to see effect)
	Middle	Resize single surface subwindow (no indication of change until release of mouse button).
	Right	no action

Note: if *map3d* does not respond as described in these tables, it could be that your window manager is grabbing the mouse/key combinations for its own purpose. 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.

Regular keyboard	
a/A-key	Switch colour tables
c-key	Toggle contour draw
d-key	Toggle depth cueing
f-key	Toggle frame lock
i-key	Toggle "direction" of color table
l-key	Toggle use of lighting
m/M-key	Step through mesh/node drawing options
n-key	Toggle display of node labels
Q-key	Quit or destroy a sub-window (Escape quits the whole program)
r-key	Reset to startup conditions
s-key	Cycle through the various surface data draw options
t-key	Toggle transformation lock
Escape	Quit the program
Clipping Controls	
<-key	Toggle front clipping plane
>-key	Toggle rear clipping plane
[/]-key	Move front clipping plane in (initially) +z/-z direction respectively
{/}-key	Move rear clipping plane in (initially) +z/-z direction respectively
1-key	Lock/Unlock clipping plane rotation with object rotation (when unlocked, shift-Middle-click rotates clipping planes)
2-key	Lock/Unlock clipping planes from each other. When active, clipping planes move together

Table 1: Keyboard controls 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. Note that when clipping is active, the screen information disappears.

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*.

Arrow Keys	
Left Arrow Key	Retreat by one frame (or current frame step)
Right Arrow Key	Advance by one frame (or current frame step)
Up Arrow key	Select next surface
Down Arrow Key	Select previous surface

Table 2: Control of *map3d* via the arrow keys

Overview of *map3d* Menus	
Mesh	display features of the mesh
Surface Data	display features of the scalar data displayed on the mesh
Scaling	links between data values and color
Contours	number or spacing and display features of the contours
Node marking	marking of the nodes
Picking	selecting times signals, mesh information, or other direct interactions with the display via the mouse
Graphics	general display features such as lighting, clipping, and depth cueing
Frame Controls	modifying frame controls
Window Attributes	features of the windows such as color and text labels

Table 3: The overall menu structure of *map3d*

.

Mesh render menu		
Render as		
	Elements	render filled surfaces for all elements
	Connectivities	render connectivity mesh
	Points	render points
	Elements and connectivity	rendered front facing triangles as elements and back facing as connectivity
Line/point size	set the size of the points from a separate window	
Color	set the color of the mesh	
Secondary Mesh Color	set color of active mesh when multiple surfaces are in same window	
Toggle mesh display	toggle display of the mesh (without changing other settings)	

Table 4: Submenus for the Mesh Display menu

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. Note that subsequent time signal picking can be set to either a) update the last time signal window to the new data channel or b) add yet another time signal window.

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

Surface Display Menus		
Color		
	Rainbow	use rainbow color map to render scalar values on the mesh
	Green to red	use green to red color map
	Black and white	use black and white color map
	Invert	invert the sense of any color map, *e.g.,* black becomes white and white becomes black
Render style		
	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
Toggle surface display	toggle display of the scalar data on the mesh (without changing other settings)	

Table 5: Submenus to control the display of scalar data on the mesh.

Scaling Menus		
Range		
	Local	scale based on the local extrema for each surface and time instant
	Global over time series	scale based on the extrema over the full times series
Function		
	Linear	linear mapping between value and color
Mapping		
	True	use true extrema

Table 6: Menu for scaling, the mapping from data value to color for rendering.

Contour Menus		
Number of Contours		
	value	selection of 5–50 contours
	Command line spacing	use the value of contour spacing in the `-cs` option of the command line
Draw style		
	Dashed line for negative values	draw positive contours in solid, negative in broken lines
	Solid lines for all contours	draw all contours in solid lines
Line size	set the line size in a separate window	
Toggle contours	toggle display of contours without changing settings	

Table 7: Menus for contour spacing/number.

There is also a menu item for adjusting the time base of the signal, manually aligning time across multiple windows, and setting the time increment between frames of data.

Node Marking Menus		
All		Make all the nodes in this (or all) surface(s)
	Sphere	mark each node with a sphere
	Node #	mark each node with the node number in the geometry
	Channel #	mark each node with the associated data channel number
	Data value	mark each node with the associated data value
	Color	set the color for marking all nodes
	Size	set the size of all node markings
	Clear all marks	remove all node marking settings
Extrema		Make all the nodes that are the extrema
	Sphere	mark each extrema with a sphere
	Node #	mark each extrema with the node number in the geometry
	Channel #	mark each extrema with the associated data channel number
	Data value	mark each extrema with the associated data value
	Size	set the size of all extrema markings
	Clear all marks	remove all extrema marking settings
Time signal		Make all the nodes that identify the location of time signals shown in the display
	Sphere	mark each times signal location with a sphere
	Node #	mark each times signal location with the node number in the geometry
	Channel #	mark each times signal location with the associated data channel number
	Data value	mark each times signal location with the associated data value
	Color	set the color for marking all times signal locations
	Size	set the size of all time signal markings
	Clear all marks	remove all time signal marking settings
Toggle node marking		toggle display of the selected markings

Table 8: Menus for marking nodes in the display. If all surfaces are currently displayed, any of these settings will affect all surfaces. If we have a single (or current) surface only, then change only that surface.

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,*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, or the “f” key, to toggle the frame lock. Any other attempt at input will not be accepted.

8.4 Picking mode

By “picking” we mean selecting some piece of the display in the current window using the mouse (with buttons). *map3d* will eventually support selection of a number of different elements of the display: nodes, triangles, scalar leads, *etc,* all of which either return some information, or affect the display, or even the geometry of the display. In version 5.0Beta, the choices are limited to node information and a time signal.

Picking Menus		
Size of picking aperture		select the size of the region around the mouse cursor that will register a "hit" when picking; larger values will make it easier to pick an object but also easier to hit multiple objects.
Time signal		alt-left mouse will select a time signal to show in a separate window
	(refresh window mode)	update the last time signal window with each pick of a node
	(new window mode)	create a new time signal window with each pick of a node

Table 9: Pick mode menus. At present, picking only selects a time signal from the surface geometry and displays that time signal in a separate window. Much new functionality will be coming soon to picking.

Graphics Menus		
Light source		select the source for the lighting model
	From above	light shines from above the surface
	None	no lighting (turn lighting off)
Toggle clipping		toggles particular clipping plane options
	Front plane	toggles front clipping plane
	Back plane	toggles rear clipping plane
	Locking planes together	makes planes translate together
	Locking planes with object	rotate surface with the planes or rotate surface through the planes
Depth cue		apply depth cueing (fog)
	Turn on	switch depth cueing on
	Turn off	switch depth cueing off

Table 10: Graphics menus. These control general graphic rendering options.

Frame Control Menus		
Lock Frames	toggle whether frames operations affect one surface or all surfaces	
Set Frame Interval		
	value	select between 1 and 90 for frame animation step
	Command Line Interval	use the value of interval specified in the -i option of the command line
Reset Frames to 0	positions the surface at the first position in time	

Table 11: Controls for the attributes of the map3d windows.

To control picking, use the top-level "Picking" menu. See Table 9 for the available options.

8.4.1 Picking Nodes

This mode simple returns the location, in the same coordinates in which the point data was originally read into the program (the points are shifted to keep the object at the center of the screen's coordinate system).

Window Attributes Menus		
Screen info		select the text written to the screen
	Turn screen info on	
	Turn screen info off	
Color		select window colors with the separate color selector
	Background	select background color for the window
	Foreground	select foreground color for the window
Size		select some size options
	Double current size	double the current window size
	Half current size	half the current window size
Toggle Transformation Lock		toggle whether surfaces transform together or independently
Save Mesh		save the current position of the surface in a .pts and a .fac file

Table 12: Controls for the attributes of the map3d windows.

9 Output from *map3d*

The only means of getting output from *map3d* at the moment is by capturing the image directly from the monitor, for example, with the Irix "snapshot" utility, or by direct photography. This will change soon...

10 BUGS

To many to even begin to contemplate. But if there are any you would like exterminated, please send email to map3d@cs.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 addressing:

- add contour lines to color scale bar; likewise for band shading
- map3d without arguments should generate the usage string to the user
- add more scaling options
- add more picking options, *e.g.,* editing the geometry
- add landmarks to the display, a feature of older versions not yet converted
- generate output images, postscript, geometry files
- remote control of the animations and user interventions for making videos
- settings files to keep your favorite settings around for the next time.
- read container files of time series data

References

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

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

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

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

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