Map3d: Scientific visualization program for multichannel time series data on unstructured, three-dimensional meshes. Program User's Guide

Robert S. MacLeod (macleod@vissgi.cvrti.utah.edu),*
Phil R. Ershler (ershler@cvrti.utah.edu),*
Christopher R. Johnson† (crj@cs.utah.edu), and
Michael A. Matheson‡

UUCS-94-016

*Nora Eccles Harrison
Cardiovascular Research and Training Institute
and
†The Department of Computer Science
at the
University of Utah
Salt Lake City, UT 84112 USA
and
‡Silicon Graphics Incorporated.

September 26, 1994

Contents

Abstract

There are many examples in biomedical research of recording time signals with multi-sensor arrays whose elements are arranged in irregular, three-dimensional grids [1, 2, 3, 4]. If the nodes in such arrays can be connected into "surfaces", it is possible to employ linear interpolation techniques to display scalar data values on the grid, provided the node locations, and their connectivities, are known. The program *map3d* was developed to provide flexible interaction with such grids, and the data values associated with them. Briefly, *map3d* provides interactive display of the nodes of the sensor array, and the connectivity mesh (described as line segments, triangles, or tetrahedra), with capabilities for manual editing of the connectivities and subsequent storage of the results. If scalar data are available associated wth the nodes of the mesh, *map3d* will display this data as either colour-coded contour lines or Gouraud or flat shaded surface elements. If there are series of such scalar data (typically in time), the time signals for selected channels can be displayed and used to select which dataset to display. The program has extensive scaling options for data value to colour mapping, different colour maps, and a variety of methods of determining the scope of data scaling. Above all else, the program allows intensive interrogation of both the geometrical mesh and the data values mapped to it.

This technical report describes the design concepts and user interface of the *map3d* program. The code was written in ANSI complient C and makes extensive use of the Graphics Library (GL) routines from Silicon Graphics. The code has been ported, in a somewhat limited edition, to the IBM RS-6000 running AIX, as well as to the Sun Sparc architecture using both Dupont Pixel's and Portable Graphics' (NPGL) GL emulation software. It does not run under OpenGL at this time. The code is not in the public domain, however, the author (macleod@cvrti.utah.edu) will consider reasonable requests for information.

1 Introduction

This document describes the function and usage of the program *map3d*, a scientific visualization application used at the CVRTI to view model geometry and scalar (potential) and vectors (current) field distributions. The program is highly interactive and makes extensive use of the full VGX features of the Silicon Graphics Library (GL). *map3d* is a program written in C using the SGI GL library for interactive display of both geometry and data assigned to elements of that geometry on the Silicon Graphics 4D/210 VGX workstation. The program can read multiple surfaces, each with multiple associated potential/current data files.

2 Usage

```
map3d [-b -iv -ss -v -w -nw\
       -c  default_mesh_colour \
       -df default_filename \
       -dp datafile_pathname \
       -gp geomfile_pathname \
       -rl report_level \
       -vw xmin ymin ]
       -f geomfilename \
       [-ac \
       -as xmin xmax ymin ymax \
       -c colour \
       -cg colour \
       -ds \
       -dp datafile_pathname \
       -gp geomfile_pathname \
       -lm landmark_filename \
       -p potfilename -s num1 num2 -i increment \
       -ph maxpotval -pl minpotval \
       -ps scaling_value \
       -ll leadlinks-filename \
       -cl channels-filename \
       -sl surfnum \
       -t trace-lead-number \
       -at xmin xmax ymin ymax]
```

2.1 Typical examples

Typical cases of usage of *map3d* are given below:

- Read data from a .data file and get the geometry filename from the header of that .data file.

  ```
  map3d -nw -f datfilename.data
  ```

  The -nw option sets a new window for each surface of data. A modification of this if the number of the data series is known in advance is

  ```
  map3d -f datfilename.data@seriesnum
  ```

  This would load the series `seriesnum` from the file `datfilename.data` and place all surfaces in a single window.

- Read data from one .data file and geometry from a specific .geom file.

  ```
  map3d -w -f geomfilename.geom -p datfilename.data
  ```

  This overrides the geometry file specified in the header of the .data file. The same modification as above is permitted here — if the specific series number is already known, use the form

  ```
  map3d -w -f geomfilename.geom -p datfilename.data@seriesnum
  ```

  This reads series `seriesnum` from `datfilename.data` and displays it on the geometry from `geomfilename.geom`.

- A third, potentially very powerful mode is permitted in which the surface number for the geometry is specified explicitly, *i.e.,*

  ```
  map3d -w -f geomfilename.geom@surfnum -p datfilename.data
  ```

  which specifies that surface `surfnum` is to be read from the geometry file and that only data associated with this surface will be read from the data file `datfilename.data`. Here too the option of explicitly specifying the series number is provided. For example,

```
map3d -w -f geomfile.geom@surfnum -p datfile.data@seriesnum
```

would read geometry from surface `surfnum` and data for that surface from series `seriesnum` of the file `datfilename.data`. In this mode, all the surface based attributes described below (*e.g.,* colours, window locations, first and last frame numbers of the data, *etc.* can be specified at startup time.

Note that if the series number is not specified at startup, the user is presented with a file box from which the desired series is selected. Likewise, if the frame numbers are not given in the command line, the user is presented with a power curve in which to window the data to be displayed. See section 4.3.2 for more information in how to access .data files.

2.2 Global Parameters

The following general parameters affect the entire display:

`-b` = open each window without borders. This provides a neater look in multiple-window displays and by using the alt-key and right mouse button, movement or re-scaling of the windows is possible after they are placed on the screen.

`-iv` = initialize video equipment in VISSGI.

`-ss` = make sure all windows have the same scaling applied.

`-v` = run program in video mode. This thickens lines, and sets a reduced range of colours, for the purpose of making videos from graphics output. This also initializes the video equipment in VISSGI.

`-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.

`-w` = for multiple surfaces (*i.e.,* more than one set of points and triangles), place all surface data in the same window. By default, *map3d* opens a new window for each new data surface. (This is the default and is hence redundant.)

`-c default_mesh_colour` = colour index to use of all surfaces for which there is no specific colour specification.

`-df filename` = read the defaults file `filename` for start-up settings for the program. This overrides any defaults files read by the program (see below) or any defaults set within the program.

`-dp datafile_pathname` = directs the search for data files accessed via a .data file to another directory. This occurs most often with data files that actually point to .pak or .raw files on the Vax. 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 specified in the defaults file (see section 4.1) or via an environmental variable called `MAP3D_DATAPATH`, which you can set at any time before executing *map3d*. You may specify this option either globally for all surfaces by inserting is before the first `-f filename` in the command line, or separately for each surface.

`-gp geomfile_pathname` = directs the search for geometry files accessed via a .data file to another directory. The most common case of this is when the name of the geometry file is stored in the data file, usually without directory specification. This value can also be specified in the defaults file (see section 4.1) or via an environmental variable called `MAP3D_GEOMPATH`, which you can set at any time before executing *map3d*. You may specify this option either globally for all surfaces by inserting is before the first `-f filename` in the command line, or separately for each surface.

`-rl report_level` = set the report level for this execution of *map3d*. The legal values are 0 (no reporting) to 3 (full debug-level reporting). This values overrides the one set in the default file, bit should be specified after the default file if the `-df` option is used.

`-vw xmin ymin` = Set the lower left hand corner of the video window to a particular spot on the screen. Units are in screen coordinates, same as for the `-as` and `-ts` commands described below.

`-f geomfile.geom or datafile.data` = filename of the .geom (geometry) or .data file containing points and connectivity information. Note: this is the minimal input for successful function of *map3d*. If a .data file is used, it is assumed to contain explicit reference to the associated .geom file containing the geometry. All surfaces are read from the geometry file.

2.3 Surface parameters

For each new surface, a subset of the following surface-specific parameters may be used. Note that if these parameters are to be used with data that are read from .data or .geom files, the last of the three forms specified in section 2.1 above must be used. Otherwise, all surfaces are read at once and the user must accept the default settings for each surface.

`-f geomfilename` = root filename of the geometry files (.pts, .fac, .tetra, .geom, or .data) for the triangularized surface to be displayed, *eg.,* `-f RL-torso` for the files `RL-torso.pts`, `RL-torso.fac`, and `RL-torso.tetra`. If a .geom file is the source of geometry, a single surface may be selected by appending the surface number N to the filename in the form "@N", *e.g.,* `geomfile.geom@3` to read surface 3 from .geom file `geomfile.geom`.

`-ac` = do not shift the origin of the points to the centroid of the pointset but leave values as they are in the input geometry files.

`-as xmin xmax ymin ymax` = set the absolute location in pixels of the surface window most recently defined. This can be used to set up multiple window displays in which the location of each window is set beforehand. The full screen of the SGI 4D has 1280 by 1024 pixels.

`-c colour` = desired colour for the mesh of a particular surface

1 = red	4 = cyan
2 = green	5 = magenta
3 = blue	6 = yellow
7 = white	

`-cg colour` = desired colour for the vectors in any grad files read for this surface. This is meaningful when the amplitudes of the vectors have no intrinsic meaning (*e.g.,* fiber-orientation vectors).

1 = red	4 = cyan
2 = green	5 = magenta
3 = blue	6 = yellow
7 = white	

`-ds` = make this surface the dominant surface in a master/slave relationship used in the program for moving surfaces independently of one another.

`-dp datafile_pathname` = same as above but for this surface only.

`-gp geomfile_pathname` = same as above but for this surface only.

`-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 4.5 below for details.

2.4 Data parameters

For each set of potentials on the surface:

`-p potfilename` = base filename for the potential and current data files. If the `-s` option is used, a number and the extensions `.pot` for potential and `.grad` for currents will be append to this base filename (see `-s` option), for .data files, the `-s` option specifies the frame numbers to be read from the file. Otherwise singles .pot or .grad files named `potfilename.pot` and `potfilename.grad` will be searched and read in, or the user will have to interactively specify the run number and frame numbers to be read from the .data file. If a specific run of a .data file is desired, append the run number N in the form "@N to the end of the filename, *e.g.,* `-p datafile.data@4` to read run number 4 from the file datafile.data (see section 2.1).

`-s num1 num2` = range of frame numbers to read. If we are reading data from .pot or .grad files, these values are appended to the value of `potfilename` to make complete pot filenames.

eg., `-p lux_map -s 1 3` expands to: `lux_map001.pot lux_map002.pot lux_map003.pot` If we are reading from a .data file, frames `num1` to `num2` are read from the file.

`-i increment` = difference between each frame number. For .pot and .grad files this affects the string appended to potfilename for each new file.
eg., `-p lux_map -s 1 6 -i 2`
expands to: `lux_map001.pot lux_map003.pot lux_map005.pot` For .data files, the frames are read with an increment of `increment` from the file.

`-ph maxpotval` = maximum data value in "user" scaling mode. The user can select between "local", "global", and "user" scaling modes using the mouse menus (see below).

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

`-ps scaleval` = scaling value multiplied by each potential value as it read in from the file(s).

`-ll leadlinks-filename` = file in *leadlinks* format containing a list of the node locations that correspond to a subset of the leads, *e.g.,* the measurement lead locations on the ventricle surface. The nodes identified by the leadlink file are displayed as cubes (or spheres), with the lead number (not necessarily the same as the node number; this is, after all, the purpose of the leadlink file in general) optional drawn beside it. See the `l` command below to toggle display of the leads in the leadlink file. See section 4.4 for more information.

`-cl 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 lead locations when lead numbers are displayed. See section 4.4 for more information.

`-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.

`-t trace-lead-number` = number of the node to be used for the display of a single scalar lead in its own window. If this is not specified, no scalar display is provided on startup. At any time after this the user can select a lead via the pick mode menu item and have the scalar data from that lead displayed.

`-at xmin xmax ymin ymax` = set the absolute location in pixels of the scalar window most recently defined. This can be used to set up multiple window displays in which the location of each window is set beforehand. The full screen of the SGI 4D has 1280 by 1024 pixels.

3 New Features

In this section, we hightlight 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.

3.1 Applying transformations to data

Background: The idea of this feature is to be able to apply the same translations and rotations that are possible within *map3d* to the geometry data, and save the result for use elsewhere. The application was to produce 2D projections from 3D meshes and save the results in a projection that made it useful for 2D plotting programs like the infamous *Z* and less famous *pscont*.

Usage: Access to this command is from the menu, and is best described in the following steps:

1. Bring up geometry and select the surface(s) to be transformed.
2. Rotate and translate until the desired project (view) of the geometry is fixed.
3. Select from the main menu, the "Rotation/Translation" submenu, then the "Apply Current Rotation to Geometry" choice.
4. This will cause only subtle visible changes on the screen, but if the axis are turned on (X-key), they will flip back to the default position , while the image of the geometry remains fixed, a sign that translations have been applied to the geometry.
5. When leaving *map3d*, make sure you check the window from which it was first started for the dialog that directs the saving of the transformed data in a new file. Choices are for with .pts/.fac or .geom output files (see section 4.2 for detail).

3.2 Command line parsing:

Background: The parsing of the command line has been completely rewritten in the latest version of *map3d*. This should not mean a great deal to look and feel of the program, but should make it more robust and predictable than was previously the case. However, until the dust all settles, please do not be surprised if things do not go quite as expected. This may be a sign of changes in the structure of the command line parsing, but also of bugs in the new version. Please document the behavior carefully and let me know.

Usage: This change has allowed two interesting features:

1. *map3d* can be started wit nothing more that a single argument which contains the name of the .data or geometry files; the -f is not necessary. Note, however, than **this is not true if more than one argument is passed**
2. *map3d* will soon support selection of the time series to be viewed from with the same invocatin of the program. Stay tuned...

3.3 Single *versus* multiple windows

At long last, I have changed the default for the -w option so that if not otherwise specified, all surfaces will appear inthe same window. TO have each surface in its own window, the argument -nw (= "new window") must be in the command line.

3.4 Landmark display

Background: The display of landmarks is now more flexible than in previous versions, with some simple lighting models added to provide more colours and views of the landmarks. There is also a new landmark which describes a clipping plane through the geometry.

Usage: To play with the display options of the landmarks, select the "Drawing Landmarks" menu option, then one or more of the lighting models and colours. The options are set up as follows:

Draw default shaded landmarks: this is the same mode as was used before.

Draw eye-point shaded landmarks: this is the simplest of light models in which the colour shading varies with angles away from a line between an imaginary eye location and the center of the object. Use this,together with the "Select eye-point location" item to give a different look to the landmarks.

Landmark shading colour: This selects the colour used to draw the landmarks. There are only a few basic colours implements at the moment, but this can be expanded at any time. Note that by interactivly altering the colour map, different colours can be tested, but only with major limitations since the colour intensity varies with the lighting model.

Draw mesh landmarks: This mode is primarily for debugging the definition files for the landmark files and shows each landmark in its elemental segments, with labelling of each segment.

No landmarks: As before, this option turns off display of the landmarks.

Draw plane: switches the display of any landmark planes in the landmark files on and off.

Plane colour: selects the colour to be used to draw the landmark plane.

For a description of the new plane landmark and how to add it to a file, see section 4.5.

4 Input files

4.1 Default settings files

The program *map3d* looks for files containing default settings for virtually all parameters that are relevant to the control of the program. The order of precedence is as follows:

1. The `-df filename` option defines the file with highest precedence default settings.
2. A file named `.map3drc` in the current directory (the one from which the application was launched) is used next.
3. If no `.map3drc` files is found in the current directory, it is searched for in the user's home directory (see the HOME environmental variable). This file has the lowest precedence and is only used of the other two options are not found.
4. The program *map3d* has a set of internal defaults which are used if there are no external default files found.

The format of the default file is as follows:

```
# comment line (ignored by map3d)
parameter = value
```

where the parameters and values are taken from the following list:

Parameter	Values	Meaning
shadingmodel	GOURAUD	Use Gouraud shading of triangles
	FLAT	Use flat shading of triangles
scale_scope	GLOBAL_SURFACE	Scaling global over each surface
	GLOBAL_FRAME	Scaling global over each frame of data
	GLOBAL_GLOBAL	Scaling global over all data
	LOCAL	Scaling local to each frame and surface
	USER	Scaling by user-supplied values (-pl -ph)
scale_model	LINEAR	Use linear scaling of contours
	LOG	Use logarithmic scaling
	EXP	Use exponential scaling
	LAB7	Use logarithmic in 7 levels scaling
	LAB13	Use logarithmic in 13 levels scaling
scale_mapping	SYMMETRIC	scale symmetrically around both side of zero
	SEPARATE	scale separately for + and - data values
	TRUE_MAP	scale from most - to most + values
color_map	RG	Use full red-to-green colour map
	RG2	Use red-and-green (2-colour) colour map
	FULL	Use blue-to-red (full) colour map
	BTW	Use black-to-white colour map
	WTB	Use white-to-black colour map
lead_marking[1]	LEADS	mark nodes with lead (channel) numbers
	NODES	mark nodes with node numbers
	VALUES	mark leads with potential values
	CUBES	mark leads with spheres/cubes
	MINMAX_LABELS	mark extrema with lead numbers
	MINMAX_CUBES	mark extrema with cubes
	SCALAR	mark the selected scalar node
num_cols	value	number of colours to use in shade display
num_conts	value	number of contour levels in display
draw_bbox	TRUE/FALSE	set bounding box on or off
datafile_path	pathame	alternate path to the .pak/.raw files in .data file.
geomfile_path	pathame	alternate path to the .geom files in .data file.
report_level	value	level of error reporting (0–3)

Note that these parameters and values are not case sensitive and that they can all be overridden during execution of the program, typically via the mouse menu (right mouse button). See section 5.2 for details on the different scaling options. The list of parameters possible will also certainly grow with the program.

[1]options are cumulative

To save the current settings in a file, there is an option in the main menu of *map3d* which dumps all the settings to the file `./.map3drc`, that is, the file `.map3drc` in the current directory. That way, when you start the program again from that directory, the same settings will be loaded. The `.map3drc` file is just a normal text file, but like all "rc"-files, it is hidden from the `ls` command unless you add the `-a` option (the alias `la` is set up by default to do a long listing of all files, including hidden files). Dumping a copy of the settings is also the easiest way to see what settings are currently being maintained by *map3d* and also forms the best starting point to setting up your own customized default settings files.

4.2 Geometry input files

The input of geometrical data occurs via the "standard" .pts, .seg, .fac, and .tetra files, as well as the new .geom geometry file format. The *map3d* program, when fed a base filename with the -f option, looks for all possible candidate geometry files and queries the user for resolution of any ambiguities.

4.2.1 .pts/.fac files

One .fac (.pts) file contains the connectivities (points) for a single surface, hence, multiple surfaces must be realized by building multiple files. If you have a .tri file containing single or multiple surfaces on the Vax, we can move it to the SGI and unpack it in such a way as to make it accessible by *map3d*. For viewing tetrahedral meshes, use the .tetra file format, together with the associated .pts files. A file in a format for viewing geometry composed of straight segments (two points connected by a line) is identified by the .seg. extension.

4.2.2 Standard geometry (`.geom`) files

The *map3d* program is capable of reading geometry data from the new geometry file format (.geom files). See other documentation for a description of this file type and how to use it. The geometry in a .geom file can be specified in the `-f` option, or found indirectly via the filename stored in a .data file (see section 4.3.2 below)

4.3 Potential and gradient data files

4.3.1 .pot and .grad files

The scalar data values that are assigned to the points in the geometry are input via `.pot` files, while vectors are input via `.grad` files. **Scalars** are ordered in the same way as the node points as a simple one-to-one assignment is performed, although a channels file can be used to provide indirection (see section 4.4) . **Vector information** is self-contained since it is expressed as start points and vector components in 3-D coordinates and may, therefore, be ordered in any way the user wishes.

The rules for `.pot` files are:

- Each line of the files contains one data value, assumed to be real.
- Order of the values must either agree with that of the associated set of points or a channels file must be supplied to redirect the links between potential value and nodes.
- File *must* end with a blank line.
- A single file can contain only the data values associated with a single surface at a single instant in time.
- The extension `.pot` *must* be used.

The rules for `.grad` files are:

- Each line of the files contains 6 ASCII strings, assumed to represent real data values as follows:
  - First three values define the starting point of the vector in three dimensions, in the same coordinate system as the points of the geometry.
  - Second set of three values represent the X-, Y-, and Z-components of the vector.
- File *must* end with a blank line.
- A single file can contain only the vectors associated with a single surface at a single instant in time.
- The extension `.grad` *must* be used.

A set of files that contains a time sequence of data should have filenames of the form base-filename001 to base-filename NNN, with the numeric suffix (NNN) being a three-digit value, which can run in any regular increments. For example, pot-file001.pot, pot-file005.pot, pot-file009.pot, pot-file013.pot, ...

4.3.2 CVRTI data (.data) format files

The newly developed standard data files (.data files) are capable of holding not only the scalar and vector data found in the older .pot and .grad files, but also links into existing "pak-tape" files, the name of the suggested geometry files, and various other bits of information that could be useful to *map3d*.

There are some concepts of the data file structure that should be understood to appreciate the different modes of operation described in section 2.1.

Series or runs By a series or run, we refer to the basic structural entity of a .data file: .data files contain a header, and then runs of data. Data within a run belong together, typically because they have been recorded at the same time. A run of data is often multiplexed, *i.e.,* it has numerous leads of data, potentially over numerous surfaces. If *map3d* is launched without information specifying the run number, a window will appear containing a list of the titles given to each of the runs of data in the .data file. The user then selects the run that should be displayed. The desired run number can also be appended to the .data filename with a "@", as in

```
map3d -f datafile.data@2
```

which selects run 2 from the .data file `datafile.data`.

Links to geometry The links between the leads of data in the .data file and the nodes of the surface[s] over which they are displayed is established via *channel* array information, which is often stored as associated scalars to the nodes of the geometry file (see documentation on Geometry and Data files elsewhere). Section 4.4 contains more information on channels and leadlinks arrays.

Frames By *frames* of data, we mean instants in the data representing single moments in time. For each frame, there is a map. If *map3d* is launched without specific information on which frames are to be displayed, a window then pops up containing the power curve for the entire run and the user sets both the start and stop frames and then clicks on "accept". To explicitly specify frame numbers, the third form described in section 2.1 must be used to launch *map3d*. For example

```
map3d -w -f geomfile.geom@1 -p datafile.data@2 -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 .data file `datafile.data`.

4.4 Leadlinks and Channels

4.4.1 Description of leadlinks and channels information

Leadlinks and channels files, and the arrays they contain, are identical in structure, **but not in function**. A program may require both, either, or neither of these, depending on the structure of the data files and geometries. The basic function of both leadlinks and channels information is to offer linkages between recording locations and the data that is to be associated with those locations. One , leadlinks, describes the connection between "leads", a measurement concept, with "nodes", and geometrical concept. The channels information, on the other hand, links the nodes to "channels" of data in a data file. Or, in more detail,

leadlinks The *leadlinks* information is primarily used to identify, and give numbers to, measurement leads within a set of nodes that make up the geometry. This can mean selecting a subset of the nodes, as would be used, for example, to identify the actual recording sites from a set node locations over which the data were interpolated. Leadlinks could also be used to renumber all the nodes of the geometry to reproduce the experimental setup.

In the leadlinks array, each entry refers, by its location in the array, to a particular lead; the array value at that location 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, then node 22 in the geometry will be displayed with a "4", not "22", when lead markings are selected in *map3d*.

leadlinknums The most recent change in the leadlinks structure has been the addition of the leadlinknums array, which has the same structure as the leadlinks array, but contains the actual lead number associated with each lead. This arrangement became necessary when there were cases of jumps in the lead numbers, for example when a lead is corrupted or damaged in recording, it is not available later, and should not be included in the list of leads of a display. Since the leadlinks array works purely by location in the array, we needed another level of indirection.

Hence we have the situation of a lead number K actually being called lead number L, pointing to node number M in the geometry. The *map3d* program now handles this additional indirection, and other programs are bound to follow suit as the need arises.

channels The *channels* information is used to relate nodes in the geometry to locations in the data file(s). For example, if we wish to find the data associated with node K in the geometry, then the value in location K of the channels array (`channels(k)`) will point to the correct channel in the data file.

Note that *channels arrays are used at the time the data is loaded into the internal data storage arrays!* The most frequent use of channels information is to unpack multiplexed data. This data come from input files with an inherently different structure in terms of geometry nodes and need to be sorted so that their spatial arrangements are known. The channels array provides the information needed to perform the sorting. An example would be data collected from multiple needles. The data are stored in a block, with no preconception of what spatial relationship individual electrodes on those needles might have to one another. These relationships can only be fixed by the geometry information of where each electrode was located and what surfaces these locations are grouped into. To untangle this mess, a separate *channels* array is used for each surface, to point to the measured data values which belong to that surface, and to determine which data value is associated with each node location.

If data and geometry nodes match one-to-one, there is no need for a channels array. In programs like *pscont*, on the other hand, a channels array is almost always a necessity because of the mapping from a three- to a two-dimensional geometrical representation. There are single points in the three-dimensional version of the geometry that have two equivalent points in the two-dimensional version of the geometry used for map display. Hence, *pscont* stores those channel filenames internally and associates then to a particular choice of plot geometry. In *map3d*, there are many cases in which channels information is not needed.

The figure 1 shows an example of lead and channels information layered one on top of the other. See the figure caption for details.

4.4.2 Source of leadlink and channel information

The source of both channels, leadlinks, and leadlinknums information can be either the geometry (`.geom`) file or the data (`.data`) file, or two explicit files, called "leadlinks" and "channels" files, which are described in the next sections below.

The .data and .geom files Information for the *channels* array is stored as an associated scalar with the data information in the standard .geom files. At present, there is no leadlinks array stored in the .geom file but this could change at an time.

The .leadlinks file A *leadlinks* file is an ASCII file, the first record of which contain 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:

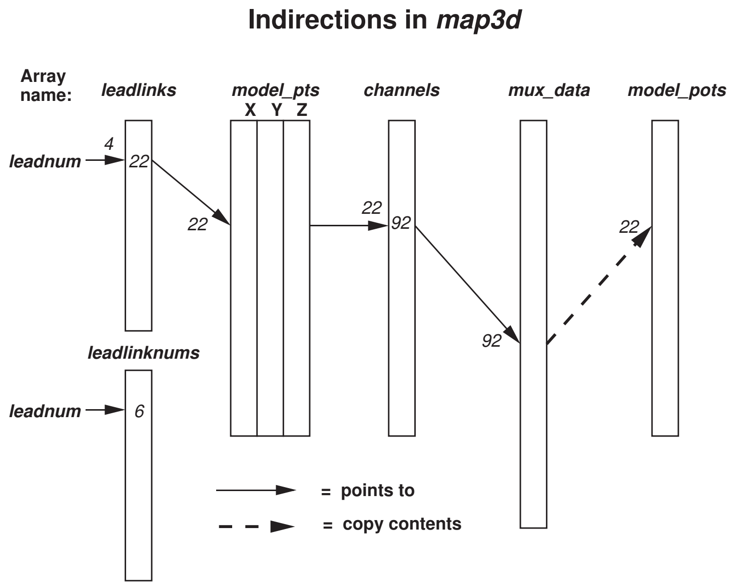

Figure 1: Example of the indirection possible in *map3d* through the use of *leadlinks leadlinknums*, and *channel* information. Lead number 4 points, via the *leadlinks* array to node number 22. This, in turn, points via the *channels* array to location 92 in the multiplexed data buffer, which causes the value at location 92 to be loaded into location 22 in the *model_pots* array.
In a separate, leadlinknums array, shown below the leadlinks array, the entry in lead 4 says that that lead should actually be called lead 6, and so any labeling of the leads should reflect this additional indirection.

1. the first number is the number of the lead, as it should appear in any labeling of the lead, the leadlinknum information.

2. the second entry in each row is the leadlink information for that lead.

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, and that lead 1 is called lead 1 and is node 1 in the geometry file. Lead two is at node 42, but 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 passed to *map3d* with the `-l` option can be marked either with coloured cubes (spheres) or labeled with the numbers of the lead or the number of the corresponding node, depending on the setting of the "Toggle node display" menu item. When we mark all the nodes, by selecting "Mark nodes with node numbers" in the "Toggle node display" menu, all nodes are labeled, not just those which are leads.

Likewise, the nodes at which the maximum and minimum potential values are found to occur, and the location used for the scalar lead display (see section 5.1) can be marked.

The `.channels` file 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). Each following record contains two integer values:

1. the first number is simply a running counter that indicates which node number the second values in the row is
2. The second element in each row is the *channel* number for that node.

For example, the file for a surface which reads:

```
352 Nodes
1    123
2    632
.    .
.    .
32   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.

4.4.3 Display of lead/channel information

With so many options for defining a node, and the lead or channel associated with it, there is room for ambiguity in the numbers that appear in a display next to the nodes. The conventions used in *map3d* are show in the following table:

Node number display conventions			
Lead Marking	**Lead/channels file present**		
	None	Channels	Leadlinks
Node numbers	same as in geometry files, starting at one for each surface	unaffected	unaffected
Lead numbers[2]	same as in geometry files, consecutive over all surfaces	channel numbers	leadnumber from leadlinks info
Data values	potential value	potential value	potential value

4.5 landmark files

The landmark files contain what I refer to as "landmarks" on the surface(s) of the geometry. Initial use was primarily for coronary arteries, but the idea has been expanded to incorporate a number of different orientation landmarks; the only requirement is that these landmarks can be described as a series of connected points, with a radius defined for each point. The 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.

A single point can also be a landmark and in this case, all that is required is a point in 3D space and a radius. The *map3d* icon for this sort of a landmark is a sphere of fixed radius and different colours, depending on the type of landmark.

The format of the landmark file is as follows:

[2]If both leadlink and channels information is present, then leadlink has priority

Line number	Contents	Comments
1	nsegs	number of landmark segments in the file
2	1 landmark_type nsegpts	segment number (1), segment type, and number of points in segment 1
3	X, Y, Z, radius	point location and radius of point 1
4	X, Y, Z, radius	point location and radius of point 2
.	.	.
.	.	.
nsegpts + 2	X, Y, Z, radius	point location and radius of last point in segment 1
.	2 landmark_type nsegpts	segment number (2), segment type, and number of points in segment 2
.	X, Y, Z, radius	point location and radius of point 1
.	X, Y, Z, radius	point location and radius of point 2
.	.	.
.	.	.
.	X, Y, Z, radius	point location and radius of last point in segment 2
.	.	.
.	.	.
.	.	.
.	.	.

The landmark types defined to date are the following:

coronary	a coronary artery segment
snare	an occlusion that can be open and closed
closure	a permanent occlusion that cannot be opened
stimulus	a stimulus site
lead	a particular electrode or lead location
plane	a cutting plane through the geometry

To specify a plane landmark requires three "points"

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

The plane is drawn as a polygon with a number of sides controlled by a program variable. See me for any changes in the way the plane is drawn.

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

5 Display features

5.1 Scalar display

When the `-t trace-lead-num` option is used, data from the incoming frames of map data are collected as a time sequence and used to construct a scalar plot of a the lead selected by the value of `trace-lead-num`. The value of `trace-lead-num` is interpreted in different ways depending on whether *leadlinks* information (see section 4.4) is present: if there is *leadlinks* information, then the lead number is passed through this indirection before being connected to a node in the geometry/data. If there is no *leadlink* information, then lead number = node number.

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 .data files contain this information.

An easier way to control the display of a scalar lead information is to let the program start without a lead selection and then choose a lead interactively. To do this,

1. Move the cursor into the window containing the surface you would like to pick a scalar lead from.
2. Select "Pick mode" from the main menu, then "Pick scalar lead" from the "Toggle Pick Action" sub-menu.
3. The cursor should now turn to a cross hairs spanning the whole screen.
4. Move the cursor over the node you would like the scalar lead from. Note that rotations, translations, and other object manipulations are all still possible even though you are in picking mode. Only the function of the left mouse button is different.
5. Select the node by clicking on the left mouse button. The cursor should change back to the arrow.
6. An outline window should appear at the cursor tip, allowing you to place and size the scalar window as desired. Move the mouse to locate the window, then anchor it with a push of the left mouse, hold the button down and drag it to size the scalar window.
7. The scalar window can now be used to select time instants and scroll through the data, as described in section 5.1.1 below.

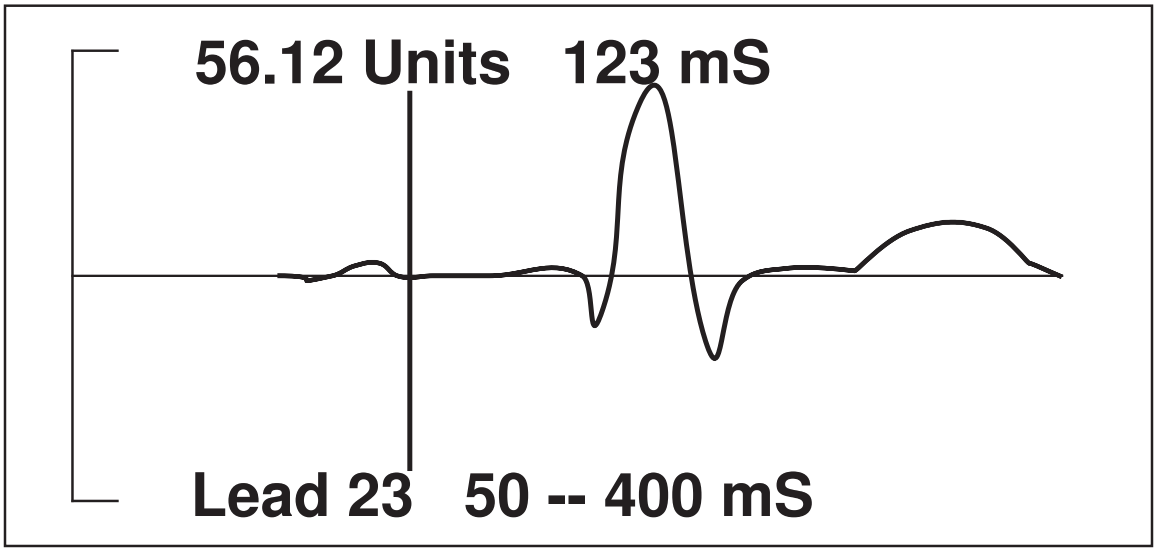

Figure 2: Scalar window layout. Use the left mouse button to grab the cursor and move it to any location on the time axis. Likewise, clicking at any point on the time axis will jump the cursor to the point, and adjust the frame number of the data displayed in the other windows.

8. Note that a single scalar window is permitted for each surface over which data is displayed.

9. Select pick scalar a second time while in pick mode, cancels the original request.

The format of the scalar display is fairly simple and self-evident, with a red vertical bar moving along the time axis as the frame number is advanced. The time axis is derived from the frame numbers of the files, not simply the number of files which are read in, *i.e.*, if frames (or pot file number) 10–20 are read in with an increment of 2, then time will begin at 10, and go through 12, 14, 16, 18 and end at 20, and will not begin at 0 or 1 and go to 5 (the number of frames of data actually read). This is also reflected in the display since the time axis always begins at 0.0; data from later in time appears shifted to the right, to a degree determined by the starting frame number.

5.1.1 Managing the scalar window

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, the map display is updated. Movement on a frame by frame basis, via the left and right arrow keys, works as it always has. The only other command allowed when the cursor is within the scalar window is the "q"-key , to shut down just the scalar window. Any other attempt at input will cause the bell to ring and an error message to be printed.

5.2 Data Scaling

Scaling is performed in a number of different ways in *map3d*, and is applied to potential and current data, both in the contour lines constructed from potential distributions, and in the colours used to display shaded and contoured potential data and current vectors. The type of scaling depends on the following parameters set by the user interactively via the mouse menus (right mouse button) or from the `.map3drc` file at startup:

Shade type This item selects the type of shading done when potential data are rendered. *Gourard* shading generates colours which represent a bilinear interpolation over each triangle area. For *flat* shading, each triangle is shaded in a single colour which represents the mean potential value of the three vertices.

 # Shades/Contours This menu choice allows the user, via a sub-menu, to set the number of shades of colour that are used for shaded renderings, or the number of lines to be used in contouring, of the potential data. The user selects amongst a number of fixed values.

Colour Map There are five different colour maps presently implemented with every chance of more to come. This menu item allows the user to select which colour map is to be used. The choices currently implemented are:

- **Red (+) to Green (-)** Largest negative value is coloured bright green, dark grays are for the region near zero, and positive values appear red.
- **Red (+) to Blue (-)** Colours range from dark blue (for negative extrema) through greens (near zero) to Red (positive extrema).
- **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.

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

Scale Scope The scope of the scaling determines which data are used to determine the extrema. In *global* scope, all the data in the entire dataset are used, even those not presently visible on the display. In *local* scaling, only the data presently visible are scanned for extrema. The *user* scaling scope uses the current user-selected values for maximum and minimum for the scaling (see `-pl` and `-ph` input parameter).

Scale Model The scale model describes the way in which data are mapped to colours (or contours). The choices are: The *linear* model simply maps the data to a range of colours in a completely linear fashion, *i.e.,* $colour = K\phi$. In *logarithmic* and *exponential* scaling models, the full range of data are mapped to the colour range in a non-linear way. The *logarithmic* scaling highlights the lower level data values at the cost of poorer resolution at the higher levels *i.e.,* $colour = A\log(\phi) + B$ and *exponential* scaling does the opposite, compressing the smaller levels and expanding the higher ones to span a wider colour range, *i.e.,* $colour = Ae^{B\phi}$. The two schemes with fixed numbers of contours, *log/7-levels* and *log/13-levels* both map the upper decade (ϕ_{max} to $\phi_{max}/10.$) of the potential data range into a fixed set of logarithmically spaced values. These values are composed of a mantissa from the standard E6 (1.0, 1.5, 2.2, 3.3, 4.7, 6.8, and 10.) and E12 (1.0, 1.2, 1.5, 1.8, 2.2, 2.7, 3.3, 3.9, 4.7, 5.6, 6.8, 8.2, and 10.) number series, and an exponent such that the largest mantissa falls into the range 1.0 to 10. Hence as long as the extrema is known, it is possible to read absolute values from the individual contour lines.

Scale Mapping There are several different ways to manage the way positive and negative data are treated in the scaling transformations in *map3d*. The *symmetric* scale mapping always sets the positive and negative extrema symmetrically: the larger (in the absolute value sense) determines both maximum and minimum data values. In the *separate* scale mapping, on the other hand, 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. The third scale mapping is the simplest, *true* mapping, in which the data are used as they come: half the colours are used for data below the mean, half for data above the mean. *True* mapping is assumed for any display of current vectors since the magnitude of the vector is used to determine its colour.

5.3 Clipping Planes

The use of clipping planes is one feature of *map3d* which is undergoing constant development. At the moment, we have the capability of setting a single cutting plane at a point perpendicular to any of the six major axis of the 3D space. Once set, a clipping plane may be interactively shifted along the axis so that different regions of the data can be visualized. A second method of clipping the data is to translate the entire dataset so that it impinges upon one of the sides of the cubic bounding space (*frustum*). By moving the data, *eg.,* forward until it meets, and is clipped by, the front plane of the frustum, then sliding a clipping plane along the same axis from the back towards the front, a slab of data space can be isolated for viewing. Both of these operations require use of the dial box (see section 6).

A further degree of freedom with the clipping planes is whether the plane moves with the data during rotation and translation of the geometry, or whether the clipping planes stays fixed (with regard to the screen) while the data rotates and shifts through it. The *Clipping Planes* menu offers the user a choice between *locked* mode, in which the clipping planes rotates *with* the geometry, and *unlocked* mode, in which data moves through the fixed plane, *i.e.,* clipping plane and geometry are unlocked.

5.4 Perspective view and depth cueing

In *map3d* the default view mode is orthogonal and there is no depth cueing. Both can, however, be switched on at the user's request. To toggle between orthogonal and perspective views, use the o-key. Currently, the modeling transformations (rotation, translation, scaling) are all reset when you switch modes, but this will hopefully be handled better in the future. Note that in perspective mode, scaling and translation can be used to change the degree of perspective in the display. By moving the object further away (with the translate dial) and then increasing the scaling, the degree of perspective is reduced, while by pulling the object closer and reducing the scaling, the opposite occurs. The use of the bounding cube (see next section) can serve as additional visual feedback on the spatial definition of the object in the display.

Depth cueing works by reducing the intensity of lines and points in the display as a function of the distance from the viewer (eye location). The distance from objects in the display to the viewer is stored in a separate "z-buffer", which has a value for each pixel in the screen. Depth cueing is useful when viewing geometries or objects which would otherwise be drawn with constant colour, but is misleading and invalid when the colour is already being used to convey other information (*e.g.,* colour-coded contour lines). Hence, use depth-cueing at your own discretion. At the moment (October 27, 1994), depth cueing is only supported for drawing points and lines in *map3d*. It can be combined with perspective view to get fairly realistic images of three-dimensional objects which are drawn as a wire mesh or as points.

To control depth cueing, use the d-key to toggle, and two menu items in the "Set Clipping Plane" menu to tune. The menu items "Z-buffer near" and "Z-buffer far" allow the lower right (clipping plane) dial to be used to move the front and back (near and far) z-buffer planes either closer to (counterclockwise dial rotation) or farther from (clockwise rotation) the viewer. The trick to using this feature effectively is to note that when the **near** z-buffer plane moves through an object, ever pixel that is in **front** of it is drawn at full intensity (no depth cueing). The **far** z-buffer plane has the opposite effect in that as it moves forward through an object, pixels located **behind** it are drawn at minimal (often black or background) intensity. The region between the two z-buffer planes is spanned by the range of colour intensities which are set in the program (currently 20 linear steps span the full range of each colour). Note that this is not the same effect as moving the clipping planes around since the change in intensity is gradual, and the result depends on which z-buffer plane is used. If the two z-buffer planes meet, the object becomes invisible (for obvious reasons), a situation which *map3d* permits, but hardly encourages.

5.5 Bounding cube

If desired, a three-dimensional bounding box can be drawn around the object(s) in the display. This box can be useful when viewing in perspective mode, and may even add a decorative touch to some images. To remove it, use the "Bounding Box" option on the menu pull-down menu. Any suggestions for enhancement of this sort of thing are welcome!

6 Control of *map3d*

6.1 Menus, keyboard mapping, mouse and dial function

Control of *map3d* is by the keyboard, mouse, and dials. Many option are available via the menus inititated 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, 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 window(s) can still be moved using the alt-key together with the right mouse button.

6.2 Feedback Reporting Level

In order to control the amount of printed feedback *map3d* provides (which is posted in the window from which the program was launched), the user can select a value between 0 and 3 from the a sub-menu of the main menu titled *Report level*. The settings and level of output are as follows:

Report Level	Types of output
0	minimal output — just changes in status
1	report extrema
2	report some warning messages
3	debug mode

6.2.1 Keyboard mapping

Each key of the regular keyboard, the function keys, and the keypad are mapped to some function of the *map3d*. Most keyboard keys serve as toggles to change between a mode being on or off, *i.e.,* "m" toggles the display of the geometry mesh. A list of the keyboard keys and their functions is shown in table 1; table 3 describes the action for each of the function and arrow keys, and table 4 the actions of the keypad keys.

6.2.2 Dial box

Perhaps the most intuitive control of *map3d* occurs via the dial box. This device allows "analog" control over any GL application that makes use of dial input by allowing the user infinite control over any control parameter linked to the dials. In *map3d*, the primary continuous interaction with the program is in placing the object in the display, choosing the viewing angle and the scale factor, and adjusting the clipping planes. Hence these functions are mapped to the dials, as described in table 2 and figure 3 below.

Note that all rotations via the dial box are relative to the screen coordinate system. This system has the x-y plane aligned with the screen, the x-axis pointing from left to right, the y-axis from bottom to top, and the z-axis pointing away from the viewer, perpendicular to the screen. One can picture the rotation dials as being attached to the positive ends of the three axis. See figure 3 for a description of this.

Regular keyboard	
A-key	Switch colour table
B-key	Toggle clipping plane
C-key	Toggle contour draw
D-key	Toggle depth cueing
E-key	Toggle backfacing visibility
F-key	Shift focus to next window
G-key	Toggle display of gradients (currents)
H-key	Set the location of the video box
I-key	Toggle between display of all or single surfaces
L-key	Toggle display of leads in *leadlinks* file
M-key	Toggle mesh draw
O-key	Toggle ortho/perspective views
P-key	Toggle point draw
Q-key	Quit picking mode (Escape quits program)
R-key	Reset to startup conditions
S-key	Toggle shaded potential data draw
T-key	Whatever Rob is currently testing
U-key	Toggle locking of clipping planes
V-key	Toggle video mode
W-key	Write an image to a file
X-key	Draw axis
Z-key	Rotate about z-axis in steps
Escape	Quit the program

Table 1: Keyboard controls in map

Dial Box		Mouse Control	
Dial 0 (Bottom left)	Scale display	Left Button	Rotation/picking
Dial 1 (Bottom right)	Clipping/z-buffer	Middle Button	Image scaling
Dial 2 (2nd row left)	Z-rotate	Right Button	Menu selection
Dial 3 (2nd row right)	Z-translate		
Dial 4 (3rd row left)	Y-rotate		
Dial 5 (3rd row right)	Y-translate		
Dial 6 (top left)	X-rotate		
Dial 7 (top right)	X-translate		

Table 2: Dial controls for *map3d*

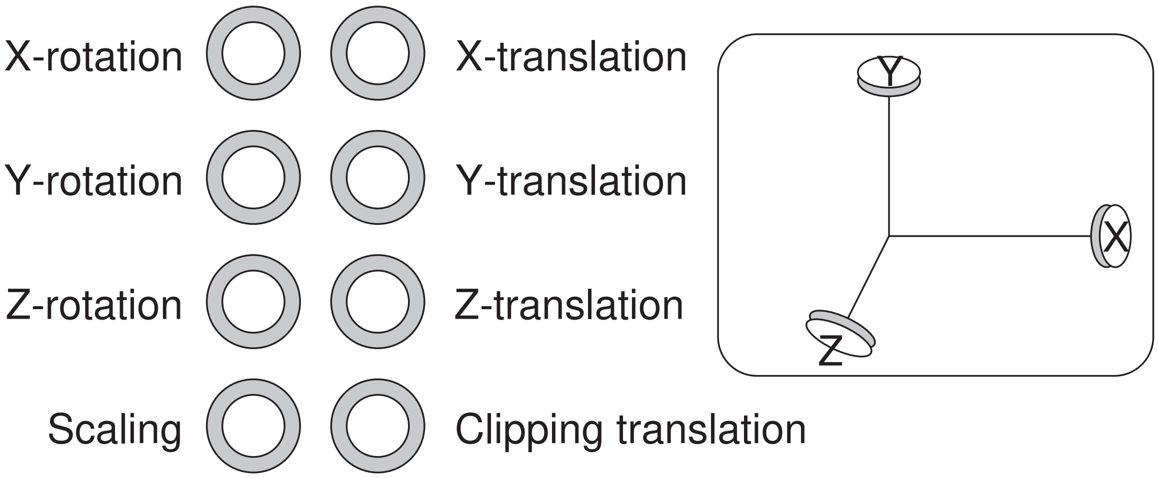

Figure 3: Dial layout for *map3d*. Rotation dials act as if they were attached to the ends of the main x, y, and z-axes, as indicated in the right hand panel of the figure.

6.2.3 Keypad Control

Providing the computer has a keypad that GL recognizes, *map3d* now also supports a keypad interface to the rotation and translations that was previously only available through the dials, and to a limited extent the mouse. See the table 4 below for details of the mapping.

6.2.4 Mouse control

The mouse can be used for different purposes depending on the current mode. This is especially important when picking is selected. Note that the middlemouse selects the most recently selected picking action, unless the “Turn off picking mode” option is selected in the “Toggle Picking” submenu. Figure 4 shows the various actions of the mouse buttons.

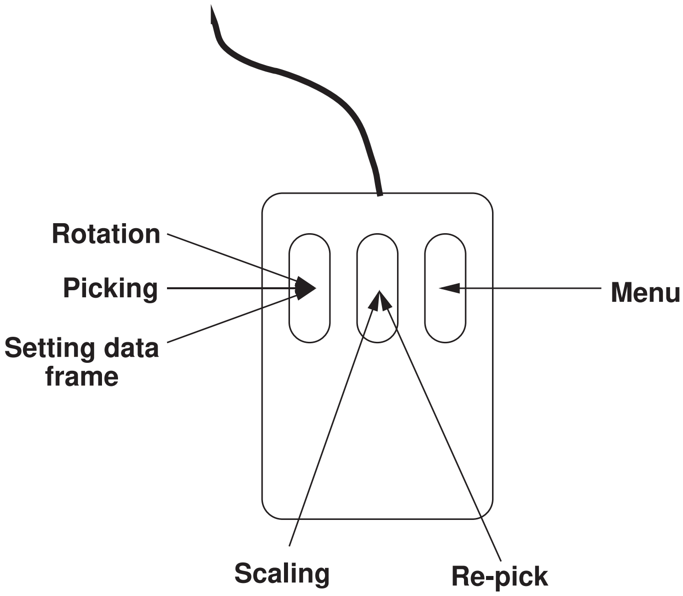

Figure 4: Mouse action for *map3d*. Mouse action of both the left and middle mouse buttons depends on the current mode. Picking makes intensive use of the mouse, as does moving the mouse into a scalar window.

When more than one surface window are used, certain manipulations apply to the window which currently has focus (the one in which the mouse cursor sits), while others may apply to all windows synchronously. Rotation and translation can apply to one or all windows (select which via the menu or F11 key), while scaling modes, colour maps, and frame numbers are applied to all windows concomitantly.

7 Picking mode

By "picking" we mean selecting some piece of the display in the current window using the mouse (with buttons). In *map3d* there are a number of different pieces that can be picked, nodes, triangles, scalar leads, *etc.,* all of which either return some information, or affect the display, or even the geometry of the display. To select what is to be picked, use the main menu and choose "Pick Mode" and then one of the choices in the "Toggle Pick Action" submenu. Below we describe each of the picking option in more detail.

Note that after picking, the program remembers what the last pick mode was and allows you to instantly go back to that pick mode by **clicking the middle mouse button**. To

Arrow Keys		Function Keys - apply to current surface	
Left Arrow Key	Rewind one image	F1-Key	Toggle point visibility
Right Arrow Key	Advance one image	F2-Key	Toggle triangle visibility
Up Arrow key	Advance one surface	F3-Key	Toggle potential visibility
Down Arrow Key	Rewind one surface	F4-Key	Toggle current vector visibility
		F5-Key	Toggle tetrahedra visibility
		F6-Key	Toggle lead display
		F11-Key	Toggle surface locking
		F12-Key	Dump image to video

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

Keypad Keys			
Keypad 4	Y-axis rotate, CW (left)	Keypad 6	Y-axis rotate, CCW (right)
Keypad 2	X-axis rotate, CCW (down)	Keypad 8	X-axis rotate, CW (up)
Keypad 7	Z-axis rotate, CCW	Keypad 9	Z-axis rotate, CW
Keypad 1	Zoom down	Keypad 3	Zoom up
Keypad 0	Shift clip plane	Keypad .	Shift clip plane
Shft-Keypad 4	-X-translation (left)	Shft-Keypad 6	+X-translation (right)
Shft-Keypad 2	-Y-translation (down)	Shft-Keypad 8	+Y-translation (up)
Shft-Keypad 7	-Z-translation (away)	Shft-Keypad 9	+Z-translation (towards)
Plus key	Increase movement	Minus Key	Decrease movement

Table 4: Keypad controls in *map3d*

have the “normal” function of the middle mouse button restored, select “Turn picking off” from the “Toggle Pick Action” submenu.

To adjust the size of the picking window, the keypad plus/minus keys can be used, much as they are to alter the sensitivity of the dials to rotation and translation. Experience suggests that a larger window makes triangulating (selecting nodes) easier while a small window makes selecting triangles (to remove or flip them). Each picking mode maintains its own picking window size so that changes affect only the current picking mode.

7.1 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).

7.2 Picking Triangles

When simple triangle picking is selected, the node numbers of each of the vertices of the triangle are displayed, along with the coordinates of the nodes at those vertices.

7.3 Triangulating

You can edit the triangulation of the mesh by selecting the “Triangulate” option from the “Toggle Pick Mode” submenu. In this mode, the left mouse is used to select the nodes that you wish connected into a triangle. Note that the cursor changes each time you select a valid point (see figure 5). If there is no change,then you either hit no points, or two points. Adjust the display, or the sensitivity of the picking window (keypad plus/minus keys) and try again. The window from which *map3d* was launched will tell you more, and on the Indigo(s), sound effects provide audio feedback.

Pushing the middle mouse button selects the nearest triangle and kills it, removing it from the list.

Files with the latest version of the connectivity are updated each time you leave triangulate mode, which you can do by either hitting the “Q” key or selecting “Turn picking off” option. The filenames are created in a random fashion and are (optional) removed when you end the session and save the latest version of the geometry in a file of you choice. When you go to exit *map3d*, you will be asked for the basefilename into which you wish the updated

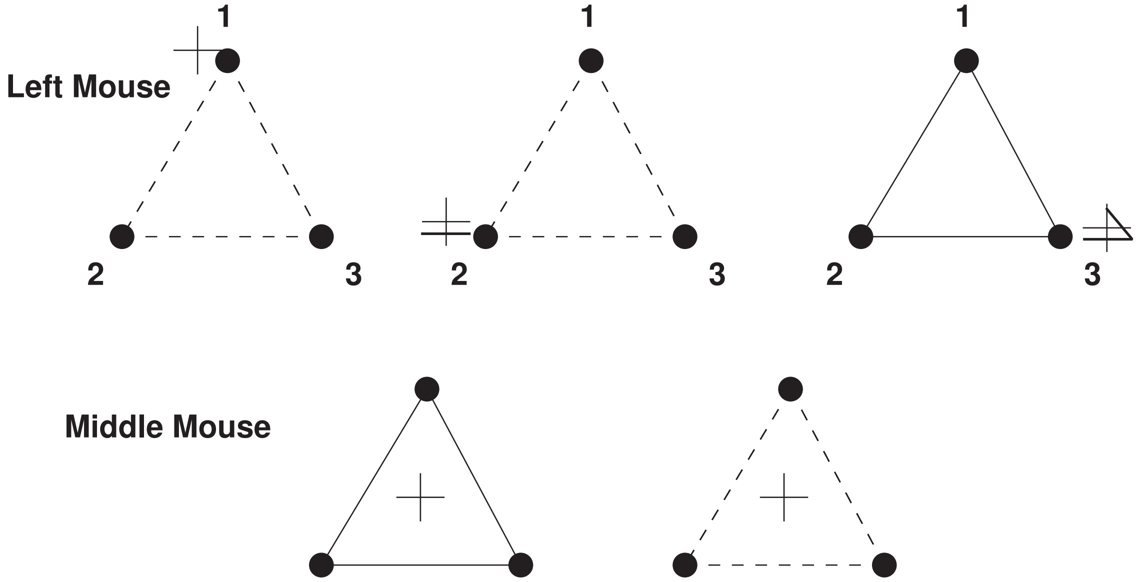

Figure 5: Triangulation modes and cursor changes. Before picking the first point of the triangle, the cursor is a short cross-hair. After successful picking of the first point, the cursor gets a horizontal mark (one side of the triangle) added. Selecting the second point adds another side to the triangle and picking the third point returns to the original cross-hair. Always pick in a counterclockwise pattern as seen from the outside of the surface for proper triangle orientation. The middle mouse can be used to remove a triangle from the mesh.

version of all surfaces (points and connectivities) to be stored. At the moment, these files are .pts/.fac files, which can easily be converted to .geom files if needed.

7.4 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 GL to decide which triangles to show in "hide backfacing triangles" mode (which you can toggle with the E-key in *map3d*). Unfortunately, it is not always possible when constructing geometric models to tell which way the triangle is to be viewed — this is still something humans do better than computers — and so we often need to edit a geometric model so that the triangles are 'flipped' the right way. Hence, the "Flip triangle" option in the "Toggle

Pick Action" submenu.

To check a model for correct triangle orientation and flip the offending triangles, select "Flip triangle" from the "Toggle Pick Action" submenu. The cursor will change shape (to a cross-hairs with triangle around it) and *map3d* will go into "hide backfacing triangles" mode. This means that only the front-facing triangles are drawn and the holes indicate which triangles need flipping. Then the cursor in each triangle you want to flip and hit the left mouse button. If you were successful, the triangle will fill in.

This option, like triangulating, produces a new connectivity file, which is saved temporarily in a random filename each time you leave triangle-flip mode. When the program finishes, you will be prompted for the base filename of a new set of points and fac files, thus preserving the original files for later use.

8 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, video tape, or film. This section describes some of the methods available for this process.

8.1 Postscript dump

With the addition of the psgl software (after significant modification), we are able to generate postscript files from map3d. There are still some restrictions and bugs in the system, but the essentials work and I encourage its use. To create a postscript file, run *map3d* and set up the view you want. Then push Control-Shift-PrntScreen, all at the same time. You will see some activity in the window you started *map3d* from, and the name of the output file will be listed somewhere in the output.

You can view the resulting postscript file with *xpsview* and print it directory using the *printit* command. The psgl software produces proper postscript files, not images dumped to postscript so the output quality is excellent.

At the moment, the system defaults to producing black lines (or gray shades) on a white page, but the option exists to produce true colour postscript. To select this, go to the Shading/Contours menu or the Colour Maps menu and select Toggle Postscript background. When you see a light background in the *map3d* window, psgl will produce true colour postscript. Otherwise, the default black lines on a white page will come out. Dashed lines are also printed by psgl, if you have previously selected the "Dashed lines for negative values" option in the Colour Map menu.

Problems and known bugs include the fact that numbers that a clipped out of the view, appear to the right had side of the image, overwriting themselves and some potentially useful stuff on the page. I am working on this.

8.2 Image capture

There are no standard provisions in GL for generating output from the images generated by *map3d*. However, using an SGI utility (the same one used in the program *snapshot*) it is, in fact, possible to capture the contents of any area of the screen and save it as a file. Once preserved, this file can be viewed later, either by itself or as part of a sequence of images, with an animation program called *movie*. The contents of the file can also be converted either into other image formats for transport to other computers or conversion to Postscript for generation of hard copy.

To capture an image using *map3d* simply set the image you want to preserve and hit the "w"-key. There will be a pause of between 5–30 s and then the workstation should beep and a line will appear in the control window telling you where the image has been stored. Filenames for image storage are generated automatically, using either the name of the geometry files (if no data is present), the name of the current potential (or gradient) file if data has been read into the program, or the filename specified with the `-if` option. 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 geometry files '`daltorso.pts` and `daltorso.fac` are being displayed, with data from the file `daltorso-pot004.pot` (which was read as the fourth frame of potential data), the first file produced would be `daltorso004-0000.rgb`. Note the `.rgb` file extension, standard for this sort of file, which is also referred to as an "r-g-b" file. If, on the other hand, the user started *map3d* with the string `-if movierun`, then the images files will be named `movierun-0001.rgb`, `movierun-0002.rgb`, *etc,*.

To view a file, or a set of files again on the screen, use the *movie* program, or the *moviemaker*. Simply call the program and append to it the name(s) of the files you want to view. To view a set of image files with a common root, *e.g.,* `daltorso000-001.rgb` to `daltorso000-010.rgb`, use the calling sequence `movie daltorso000-*.rgb`. The result is an animated sequence, which can be stepped through, looped, or 'swung' back and forth. Once the images are loaded, control of *movie* is with the right-hand mouse button, pull-down menu. (See the man page on *movie*)

To convert `.rgb` files to Postscript for printing, execute the following sequence of commands:

1. Locate the directory in which the `.rgb` files have been stored. This will be the same

directory from which you started *map3d*. Use the `cd` command to move around directories and note that your prompt tells you where you are (the command `pwd` reveals this same information, should you have a non-standard prompt).

2. For each file you have saved, convert the color information to gray-scales with:

```
tobw filename.rgb filename.bw
```

   The file `filename.bw` now contains the same information as the original, but with colors removed, or converted to gray scales.

3. Now convert the file to Postscript, using the command:

```
tops filename.bw > filename.ps
```

   Note the `>` symbol, which redirects the output of the `tops` program to a file. Leaving it out will send the entire Postscript file to your display!

4. The file `filename.ps` can now be printed to a Postscript printer, using the following command:

```
printit filename.ps
```

5. After the file has been printed, unless you wish to view the files again, remove the interim files that you created, since they are quite large and consume disk space quickly:

```
rm filename.rgb
rm filename.bw
rm filename.ps
```

8.3 Video output from *map3d*

With the recent addition of a frame buffer card in the SGI, and the video local area network (V-LAN) hardware, we can now generate animation sequences on video tape from *map3d*. Control of the sequence, which may be a geometrical movement of objects in the display, or a series of sets of data on a stationary geometry (or both concurrently), is by the program *animator* described elsewhere. In effect, *animator* submits commands to the event loop of *map3d* and then captures the images from the GL window, moves them to the frame buffer, and then to video tape.

The most obvious feature of video recording in *map3d* is that the user can (actually must) select a window somewhere on the screen that defines what portion of the display is actually captured for video. If the -v option is used to launch the program, the user will see a red, rectangular box, together with a smaller, white text window with instructions. When the left mouse button is held down, moving the mouse now moves the rectangular video box; when the middle mouse button is pushed the location of the video box is fixed. The region within the innermost rectangle fairly accurately represents what will make it to the video screen. To relocate the video outline, return to the white text window and click the left mount button.

8.4 Additional video controls

With the separation of animation from video output and the desire to have thicker lines for photos without the video colour scheme, *map3d* now offers independent control of the various aspects of what was formerly video control. There is now a menu for video control, which includes options to turn the video window on (similar to the H-key), to toggle video colours (similar to the V-key) and to set the linewidth that is used.

8.5 Photographing from the Display

There are now too different ways to convert images from *map3d* (or any other output on the screen of the SGI terminal) into photographs or slides. The first, and most direct, is to set the camera up in front of the screen and blast away. The second involves grabbing images from the screen, converting them to a form understood by a Macintosh and then reading (and editing) them with an appropriate Mac program.

8.5.1 Direct photography from the screen

This process is simple and actually works very well (well enough to make the cover of journals) if you proceed as follows:

1. Use a tripod and the motor driven Canon cameras (we have two identical cameras for this purpose).
2. Square the display screen so that it is parallel to the wall, turn the intensity up to near maximum, and clean the paw prints from the glass with glass cleanser.

3. Either use the remote plunger or the delayed self-exposure mechanism to work the shutter. Otherwise vibrations from your touching the shutter will blur the images. Select the 2-second delay for minimal waiting time between shots.

4. Most pictures can be taken at the 2-second exposure time. Only with very bright images is it necessary to reduce the time and I have never needed a longer time.

5. Select manual operation of the camera — the light meter will be correct only for the case of a relatively bright, evenly illuminated image.

6. Set the f-stop at least one full f-stop larger (smaller aperture) than what the light meter suggests for anything but bright images that are very consistently illuminated. I usually take three exposures starting at this setting and moving in full-f-stop steps toward larger f-stop values. In the case of contour line images, this usually means starting at 3.5 and shooting again at 5.6 and 8.3. For shaded images, the f-stop will be larger to start with and will depend on the colour scheme used.

Some tips that I have found useful for making photos are as follows:

- You can keep a low level of room background light during photography from the screen; complete darkness is **not** required, just avoid glare off the surface of the display. I usually keep the lights from the opposite wall of the graphics lab on at a low level, enough to see my way around during the shooting.

- If you have the choice, keep the image towards the center of the screen and do not let it fill the entire screen. Also, avoid having straight lines near the edge of the image as they will appear curved in the resulting photo (turn the bounding box off).

- Select a black background for the screen (under the root menu of the windowing system, or the ToolChest/Windows/Set Background menu selection). If this doesn't seem to work, enter the command:

```
xsetroot -solid black
```

- Iconify all other windows or at least shrink them and move them out of sight.

- Select borderless windows for map3d (-b option) to get rid of ugly window borders. Borderless windows can still be moved and resized by holding the Alt-key down and pressing the right mouse button while the cursor is in the window you want to alter.

- Be imaginative — map3d permits lots of kludging and the results are worth the effort.

8.5.2 Photography via the Macintosh

The method here is very similar to the image capture and printing described in section 8.2, except that instead of converting the image to Postscript, we convert it to a form that the Macintosh can read. The form we have used with best success so far is the "pict" file, a native Apple format that can be read and written by most Macintosh applications. To convert the .rgb file `filename.rgb` produced by snapshot into the pict file `filename.pict`, enter

```
topict filename.rgb filename.pict
```

Now the file is in a form which is directly readable by the Macintosh. The file still resides on the SGI but can either be copied to the Mac using ftp (accessible from either *NCSA Telnet* or *Versaterm*), or directly read if your Macintosh is running NFS client service. Currently the Macintosh in the Graphics Lab is running NFS; to make the link, launch the NFS application and supply your own userid and password. Make sure to drag the SGI volumes to the trash after you are finished as the next person who uses the Mac would otherwise have full access to all your SGI files.

For more information on the hard- and software in the Graphics Lab, talk to Phil or see the documentation he is preparing on these facilities.

References

[1] R.C. Barr, T.M. Gallie, and M.S. Spach. Automated production of contour maps for electrophysiology II. Triangularization, verification, and organization of the geometric model. *Comp & Biom Res*, 13:154–170, 1980.

[2] P.R. Ershler, R.L. Lux, and B.W. Steadman. A 128 lead online intraoperative mapping system. In *IEEE Engineering in Biology and Medicine Society 8th Annual International Conference*, pages 1289–1291. IEEE Press, 1986.

[3] R.E. Ideker, W.M. Smith, P. Wolf, N.D. Danieley, and F.R. Bartram. Simultaneous multichannel cardiac mapping systems. *Pace*, 10:281–291, 1987.

[4] R.S. MacLeod, B.K. Hoyt, P.J. MacInnis, R.V. Potter, and B.M. Horáček. A body surface potential mapping unit for recording during coronary angioplasty. In *IEEE Engineering in Medicine and Biology Society 10th Annual International Conference*, pages 97–98. IEEE Press, 1988.