

NCSA MinMaxer

User's and Programmer's Guide

Version 1.04
30 October 1992

University of Illinois at Urbana-Champaign

Document Revision History

Product Release and Comment	Date	Document Revision
Versions 1.0 through 1.03		Online README files
Version 1.04	28 Jun 93	1.1

Copyright Software Developer

Copyright © 1992, 1993 by the Board of Trustees of the University of Illinois
 The MinMaxer software was written by Roman Waupotitsch in the Software Development Group (SDG) of the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC).

Policy

University of Illinois (UI)
 National Center for Supercomputing Applications (NCSA)
 Software Distribution Policy for Copyrighted Software
 The NCSA MinMaxer software, both binary and source, is copyrighted but available without fee for education, academic research, and non-commercial purposes. The software is copyrighted in the name of the UI, and ownership of the software remains with the UI. Users may distribute the binary and source code to third parties provided the copyright notice and this statement appear on all copies and that no charge is made for such copies. Any entity wishing to integrate all or part of the source code into a product for commercial use or resale should contact the University of Illinois, c/o NCSA, to negotiate an appropriate license for such commercial use.

Disclaimer

The University of Illinois (UI) makes no representations about the suitability of the software for any purpose. It is provided "as is" without express or implied warranty. UI shall not be liable for any damages suffered by the user of this software.

User Obligations

By copying this program, the user agrees to abide by the copyright conditions and understandings with respect to any software which is marked with a copyright notice.
 The software may have been developed under agreements between UI and the Federal Government which entitle the Government to certain rights.

Trademarks

All brand and product names are trademarks or registered trademarks of their respective holder(s).

NCSA Contacts

Send user feedback, bug reports, questions, and software and manual suggestions to the following U.S. mail or electronic mail addresses:

Send all U.S. mail to: Software Development Group NCSA MinMaxer 152 Computing Applications Bldg. 605 E. Springfield Ave. Champaign, IL 61820-5518	Email comments, questions, and suggestions to: softdev@ncsa.uiuc.edu softdev@ncsavmsa.bitnet
	Email bug reports to: bugs@ncsa.uiuc.edu bugs@ncsavmsa.bitnet

Orders

All NCSA products are available without charge from NCSA's anonymous FTP server:
 ftp ftp.ncsa.uiuc.edu
 Hardcopy manuals and software disks and tapes can be ordered through the NCSA *Technical Resources Catalog*. All orders must be prepaid. For a copy of the catalog, contact NCSA Orders by email at orders@ncsa.uiuc.edu, by phone at (217) 244-4130, or by standard mail at:

NCSA Orders
 152 Computing Applications Bldg.
 605 E. Springfield Ave.
 Champaign, IL 61820-5518

Acknowledgments

The software developer thanks Prof. Herbert Edelsbrunner, whose initial idea it was to implement MinMaxer, for his continuous support and Ernst Mücke for his help and advice concerning the graphics interface.

NCSA MinMaxer

User's and Programmer's Guide

Introduction

Chapter Overview	1
About NCSA MinMaxer	1
Features	2
Delaunay (Lawson Flip)	
Delaunay (Incremental Flip)	
Regular (Incremental Flip)	
MinMax Angle	
MaxMin Height	
MinMax Slope	
Initial (Plane Sweep)	
Notes about the Algorithms	
Differences among Versions 1.0, 1.02, 1.03, and 1.04.....	3
System Requirements.....	4
Document Overview	4
Conventions Used in This Guide	5

Chapter 1 User Interface

Chapter Overview	7
Data Format.....	7
Using MinMaxer	8
The Graphic and ASCII User Interfaces	9
Remove Duplicate Points	
Initial (Plane Sweep)	
Algorithm Selection	
Image Manipulation	
Shading	
Output Triangulations	
Interrupt and Exit	
Special Notes	13

Chapter 2 Source Code Integration

Chapter Overview	15
Resources	15
Source Code	
Sample Files	
Copyright Requirements	

Implementation	16
Data Formats	
Data Translation	
Data Manipulation	
Data Storage	
Appendix Installation and Online Structure	
Acquiring the Latest Version	19
Installation.....	20
Directory and File Structure.....	20
References	23

Figures and Tables

Figure I.1	Sample screen listing	6
Figure 1.1	The MinMaxer interface in non-graphic mode	9
Figure 1.2	The MinMaxer control panel in graphic mode	10
Figure 1.3	Control panel insert visible while triangulation is calculated	12
Figure A.1	MinMaxer Version 1.04 installed directory structure and files	21
Table I.1	Meaning of entry format notations	6
Table 2.1	Algorithms and functions	17

NCSA MinMaxer

User's and Programmer's Guide

I

Introduction

Chapter Overview

This introduction to the *NCSA MinMaxer User's and Programmer's Guide* contains the following information:

- Description of MinMaxer's features
- History of feature evolution through various releases
- System requirements
- Guide structure and contents
- Conventions used in the guide

About NCSA MinMaxer

NCSA MinMaxer is a two-dimensional triangulation tool with an optional graphic user interface. The program implements several optimal two-dimensional triangulation algorithms and can be used to aid grid generation and visualization.

MinMaxer was originally designed for educational purposes and to study the efficiency of algorithms based on the edge-insertion paradigm for computing optimal two-dimensional triangulations. MinMaxer implements several versions of the edge-insertion paradigm and contains code for the edge-flip heuristics that can be used to obtain locally optimal triangulations. It can also be used for a comparative study of the quality achieved by various other triangulations introduced in the literature.

The intent of the edge-insertion paradigm is to find the triangulation that minimizes the maximum measure over all possible triangulations of a given point set. Such measures include the largest angle or the slope of a triangle. The algorithms employed to implement the edge-insertion paradigm start with an arbitrary triangulation and iterate until no further improvement is possible. A single iteration adds a new edge to the triangulation; all edges that intersect this new edge must then be deleted. The resulting polygons are then retriangulated.

The implementation lexicographically optimizes the entire vector of measures, not just the worst one. Because of this

property, MinMaxer usually computes a unique optimum. MinMaxer uses integer arithmetic with ad hoc tie-breaking rules and simulation of simplicity (SoS) for the geometric primitives. The triangulations are stored using the quadedge data structure.

Features

NCSA MinMaxer computes optimal triangulations according to the following algorithms:

- Delaunay
- Regular
- Minmax angle
- Maxmin height
- Minmax slope
- Plane sweep

See the “References” section at the end of this guide for publications that provide complete technical information on each triangulation.

Delaunay (Lawson Flip)

For every triangle in this triangulation, the circumscribed circle through its vertices encloses no points of the set. Delaunay triangulations in two dimensions maximize minimum angles. Lawson's algorithm requires an initial triangulation and performs edge flips. This implementation uses SoS. (Delaunay 1934)

Delaunay (Incremental Flip)

This algorithm does not require an initial triangulation; it adds one point at a time and performs edge flips in the neighborhood of the new point. The triangle that contains the new point is determined by searching through a DAG (direct acyclic graph) structure. This implementation uses SoS. (Guibas, Knuth, and Sharir 1990)

Regular (Incremental Flip)

This triangulation can be described as a weighted Delaunay triangulation. If a vertex has higher weight, it tends to have more connections to its neighbor vertices. It is constructed by a slight modification of the incremental flip algorithm for Delaunay triangulations. This implementation uses SoS. (Edelsbrunner and Shah 1992)

MinMax Angle

This triangulation minimizes maximum angles. It requires an initial triangulation which is changed by a sequence of so-called edge insertions. It is usually faster if the initial triangulation has reasonably shaped triangles, such as the Delaunay triangulation. This implementation uses ad hoc tie-breaking rules. (Edelsbrunner, Tan, and Waupotitsch 1992)

MaxMin Height	This triangulation maximizes the minimum height, where the height is the height of a triangle. It requires an initial triangulation and works by edge insertions, just like the algorithm for minmax angle triangulations. This implementation uses ad hoc tie-breaking rules. (Bern et al. 1992)
MinMax Slope	This triangulation lifts the two-dimensional triangulation to a surface in three-dimensions. This triangulation minimizes the maximum slope, where slope measures the steepness in the third dimension. It requires an initial triangulation and works by edge insertions. The algorithm is usually much faster if it starts with the Delaunay triangulation of the points. This implementation uses ad hoc tie-breaking rules. (Bern et al. 1992)
Initial (Plane Sweep)	This is a fast initial triangulation by plane sweeping. The quality of the triangles is typically bad. This implementation uses SoS. (Edelsbrunner 1987)
Note about the Algorithms	<p>The MinMaxer algorithms actually achieve optimum triangulations; they are not heuristic.</p> <p>Some of the algorithms are implemented twice, with a faster solution and a slower solution. The first entry always provides the faster solution.</p>

Differences among Versions 1.0, 1.02, 1.03, and 1.04

MinMaxer Version 1.0 was released in June 1992.

Version 1.02, released in August 1992, included the following enhancements and a bugfix:

- An improved graphic interface
- A corrected version of the lighting model
- The ability to interrupt an algorithm
- The ability to turn off the visual update of edge operations
- A fix for the bug that caused convex hull edges to be preceded by 000 in a Version 1.0 output file

The algorithm implementations were not changed, so triangulations computed by Versions 1.0 and 1.02 should be identical.

Version 1.03 included a minor modification of the triangulation code and Version 1.04 corrected an error in the computation of the triangle list, but the triangulations should be identical to those computed by Version 1.0. The Version 1.04 correction affects only the triangle lists used to export triangulations.

System Requirements

NCSA MinMaxer will run in graphic mode on any Silicon Graphics workstation with IRIS 4.0.1 or later.

NCSA MinMaxer will run in non-graphic mode on most UNIX workstations.

Executables are provided for Sun and Silicon Graphics workstations.

Document Overview

This guide, the *NCSA MinMaxer User's and Programmer's Guide*, contains the following information:

Introduction

- A description to MinMaxer's features
- A history of how the features have evolved through the various releases
- System requirements
- An outline of this guide's contents
- The conventions used in this guide

Chapter 1, User Interface

- A description of the data format
- Descriptions of the user interfaces
- Special considerations and bugs

Chapter 2, Source Code Integration

- Information for integrating MinMaxer source code with an application

Appendix, Installation and Online Structure

- Instructions for acquiring the latest version of NCSA MinMaxer
- Instructions for installing MinMaxer
- The online directory and file structure for Version 1.04

References

- A list of publications that provide complete technical discussions of the algorithms used in MinMaxer

The front matter, immediately following the title page, includes instructions for contacting NCSA and the Software Development Group, information on how to acquire NCSA software, the conditions under which a user may use NCSA software, the requirements if NCSA software is to be incorporated in either commercial or non-commercial software for distribution, and other supporting information.

Conventions Used in This Guide

Most of the descriptive text in this guide is printed in 10 point New Century Schoolbook. Other typefaces have specific meanings that will help the reader understand the functionality being described.

New concepts are sometimes presented in italics on their first occurrence to indicate that they are defined within the paragraph.

Cross references within the guide usually include the title of the referenced section or chapter enclosed in quotation marks. (E.g., See Chapter 1, "User Interface," for a description of the MinMaxer user interface.)

References to documents italicize the title of the document. (E.g., See the *NCSA MinMaxer User's and Programmer's Guide* for a description of the MinMaxer user interface.)

Literal expressions and *variables* often appear in the discussion. Literal expressions are presented in Courier while variables are presented in italic Courier. A literal expression is any expression that would be entered exactly as presented, e.g., commands, command options, literal strings, and data. A variable is an expression that serves as a place holder for some other text that would be entered. Consider the expression `cp file1 file2`. `cp` is a command name and would be entered exactly as it appears, so it is printed in Courier. But `file1` and `file2` are variables, place holders for the names of real files, so they are printed in italic Courier; the user would enter the actual filenames.

This guide frequently offers sample *command lines*. Sometimes these are examples of what might be done; other times they are specific instructions to the user. Command lines may appear within running text, as in the preceding paragraph, or on a separate line, as follows:

```
cp file1 file2
```

Command lines always include one or more literal expressions and may include one or more variables, so they are printed in Courier and italic Courier as described above.

Keys that are labeled with more than one character, such as the RETURN key, are identified with all uppercase letters. Keys that are to be pressed simultaneously or in succession are linked with a hyphen. For example, "press CONTROL-A" means to press the CONTROL key then, without releasing the CONTROL key, press the A key. Similarly, "press CONTROL-SHIFT-A" means to press the CONTROL and SHIFT keys then, without releasing either of those, press the A key.

Table I.1 summarizes the use of typefaces in the technical discussion (i.e., everything except references and cross references).

Table I.1 Meaning of entry format notations

Type	Appearance	Example	Entry Method
Literal expression (commands, literal strings, data)	Courier	dothis	Enter the expression exactly as it appears.
Variables	Italic Courier	<i>filename</i>	Enter the name of the file or the specific data that this expression represents.
Special keys	Uppercase	RETURN	Press the key indicated.
Key combinations	Uppercase with hyphens between key names	CONTROL-A	While holding down the first one or two keys, press the last key.

Program listings and *screen listings* are presented in a boxed display in Courier type such as in Figure I.1, "Sample Screen Listing." When the listing is intended as a sample that the reader will use for an exercise or model, variables that the reader will change are printed in italic Courier.

Figure I.1 Sample screen listing

```

mars_53% ls -F
MinMaxer/                net.source
mars_54% cd MinMaxer
mars_55% ls -F
list.MinMaxer            minmaxer.v1.04/
mars_56% cd minmaxer.v1.04
mars_57% ls -F
COPYRIGHT                minmaxer.bin/          source.minmaxer/
README                   sample/                 source.triangulation/
mars_58%

```

Chapter 1

User Interface

Chapter Overview

This chapter discusses the following topics:

- The MinMaxer data format
- The MinMaxer graphic and non-graphic user interfaces
- Special considerations and bugs

Data Format

An NCSA MinMaxer data file contains the coordinates for the vertices of a single figure. Each line in the file contains the coordinates for one data point (a vertex) on the surface:

```
site x-coord y-coord
or
site x-coord y-coord z-coord
```

The term `site` at the beginning of each line is optional; it has no effect on the computations. The coordinates must be integers. The following lines are each valid records in a data file:

```
site    -865    -6292    686704
site    3346    -5425
1841    5839    143491
10952   5478
```

While the above are all valid records, they cannot appear in the same file. Do not mix two-dimensional and three-dimensional data; if any data point in a given data set is two-dimensional, they must all be two-dimensional. Similarly, do not mix lines with and without the term `site` in the same file.

MinMaxer works with three-dimensional input data, but does not require it. The third coordinate, the z -coordinate, is used only for weights in the regular triangulation or to calculate the slope in the minmax slope triangulation. Points with the same x - and y -coordinates are considered duplicates. If no z -coordinate appears, it is assumed to be 0 (zero).

Degeneracies are eliminated by using the SoS (simulation of simplicity) method (Edelsbrunner and Mücke, 1990) and ad hoc methods which break ties based on the indices of the vertices. Since the vertex indices are implicitly determined, the order of the input data can affect the output.

MinMaxer works with one data file at a time.

See the file `sample.data` for a sample data set.

**Running MinMaxer
with a Sample Data Set**

To run MinMaxer with a sample data set, first verify that the MinMaxer program and the file `sample.data` are available in your current working directory. (If necessary, see the Appendix, "Installation and Online Structure.") Then execute the following command:

```
MinMaxer sample.data
```

Duplicate data points confuse some of the algorithms, so select

```
Remove Duplicate Points
```

Some of the algorithms require an initial triangulation, so select

```
Initial (Plane Sweep)
```

You are now ready to experiment with the triangulations. If you are in graphic mode, you may have to hit the PAUSE key to start each algorithm.

Using MinMaxer

NCSA MinMaxer includes both graphic and non-graphic user interfaces. MinMaxer will run in non-graphic mode in most UNIX environments; the graphic interface is available only on Silicon Graphics workstations running IRIS 4.0.1 or later.

The graphic interface* presents a graphic control panel, a visual plot of the data set, a visual image of the surface geometry, and a visualization of the progress of the algorithms as they are operating. The graphic interface displays the progress of the iteration by showing edge-insertions and deletions in the wireframe illustration. Several control panel selections are available to control the calculations and the visual presentation of their progress. During the computation or after the calculations are complete, the user can select options that color the triangles according to several measures. With these last selections, one can get a global impression of how much and how fast the quality of the triangulation improves during the iteration. Using the mouse, the user can also rotate the image and zoom in or out.

In non-graphic mode, MinMaxer provides an ASCII interface and includes no visualization capabilities.

**The MinMaxer
Command Line**

The `MinMaxer` command has two options, `-g` for graphic mode and `-ng` for non-graphic mode. `-g` is the default on an SGI workstation; `-ng` is the default in all other environments.

* The graphic interface uses the GL graphics library on a Personal Iris Workstation.

If you wish to run MinMaxer while reading the following sections, execute this command after verifying that both the command and the file `sample.data` are available in your directory:

```
MinMaxer sample.data
```

MinMaxer requires that the user specify a data set on the command line, so the full syntax is

```
MinMaxer [-g|-ng] datafile
```

where `datafile` is the file containing the data set to be analyzed.

The ASCII menu in Figure 1.1 will appear when you execute the command

```
MinMaxer datafile
```

on a non-Silicon Graphics system or when you execute the command

```
MinMaxer -ng datafile
```

on a Silicon Graphics system.

Figure 1.1 The MinMaxer interface in non-graphic mode

```

bigbooty.ncsa.uiuc.edu.
_78%
_78%
_78%
_78%
_78% minmaxer.sgi -ng nice.regular.data
no graphic mode.

MINMAXer V1.03 1992

Input data is three dimensional.

          Triangulations

1 ... Initial (Plane Sweep)      6 ... MaxMin Height (n^3)
2 ... Delaunay (Lawson)          7 ... MinMax Slope (n^3)
3 ... MinMax Angle (n^2 log n)   8 ... Delaunay (Incremental)
4 ... MinMax Angle (n^3)        9 ... Regular (Incremental)
5 ... MaxMin Height (n^2 log n) 10 ... Verify Delaunayness
11 ... Verify Regularity         12 ... Remove Duplicate Points
13 ... Save Triangulation        0 ... QUIT

Make a Choice: █

```

The control panel in Figure 1.2 will appear when you execute the command

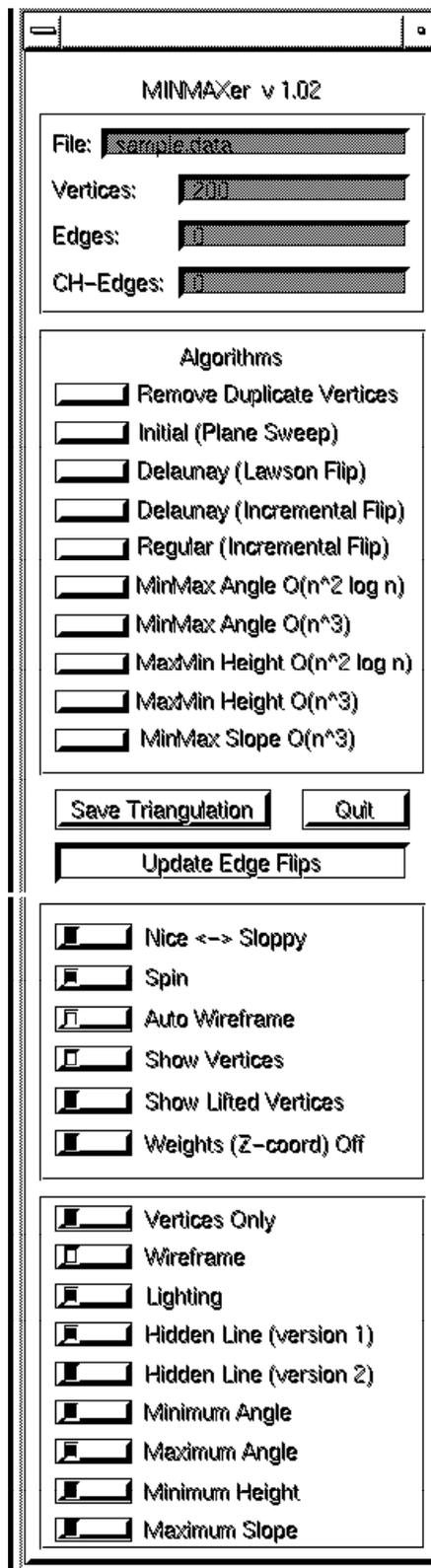
```
MinMaxer datafile
```

on a Silicon Graphics system. The control panel insert illustrated in Figure 1.3 will appear while a triangulation is being calculated in graphic mode.

The Graphic and ASCII User Interfaces

The following sections discuss several selections that are available in both the graphic and non-graphic modes and some that are available only in graphic mode. Some selections that are self-explanatory are not described.

Figure 1.2 The MinMaxer control panel in graphic mode



Remove Duplicate Points

Only the incremental algorithms are sure to succeed if the data set contains duplicate points (two or more points with the same x - and y -coordinates). Other algorithms may fail if they encounter duplicate points, so you should select

Remove Duplicate Points

before executing non-incremental algorithms.

Initial (Plane Sweep)

Several algorithms* require an initial triangulation. This can be computed fastest with the plane-sweep algorithm. Select

Initial (Plane Sweep)

While the plane sweep algorithm, by itself, produces only a marginally useful triangulation, users are encouraged to use it to initialize every data set to avoid difficulties with the other triangulations. For example, if the Lawson Flip algorithm is applied to a data set that has not been initialized, it returns an empty triangulation.

Algorithm Selection

Once you have eliminated duplicate points and performed the initial triangulation, the following triangulations are available:

Initial (Plane Sweep)
 Delaunay (Lawson Flip)
 Delaunay (Incremental Flip)
 Regular (Incremental Flip)
 MinMax Angle $O(n^2 \log n)$
 MinMax Angle $O(n^3)$
 MaxMin Height $O(n^2 \log n)$
 MaxMin Height $O(n^3)$
 MinMax Slope $O(n^3)$

As mentioned in the introduction to this guide, several triangulations are implemented with both faster and slower versions. (The faster version is always listed first.) The faster and slower versions have yielded identical results in hundreds of tests; they are provided for those who wish to verify the results.

The Delaunay triangulation is also implemented in two algorithms: the Lawson flip and the incremental flip. The choice depends on the memory available and on the data. For example, the incremental Delaunay algorithm is faster when less than 1% of the points in the data set lie on the convex hull. But, it is very memory intensive; a data set of 20,000 points requires approximately 30Mb of RAM.

The minmax and maxmin algorithms are faster when applied to the Delaunay triangulation.

* Only the regular triangulation and the incremental version of the Delaunay triangulation can be executed without an initial triangulation.

Image Manipulation

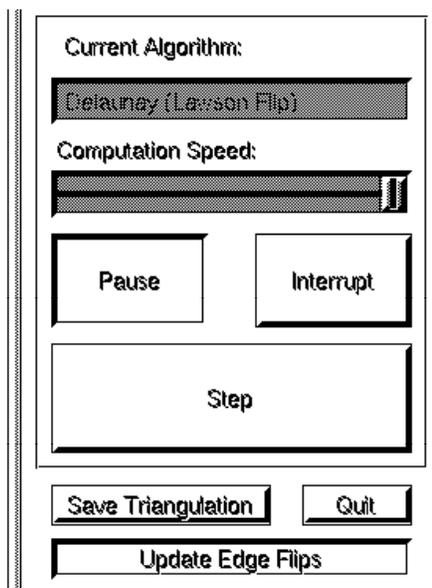
Several image manipulation selections are available while an algorithm is working in the graphic mode. Some affect the calculation of the triangulations, others affect the view presented to the user:

Calculation Selections:

These selections are presented on the temporary control panel insert that appears only while a triangulation is being calculated (see Figure 1.3):

- Pause Stops the edge computation and allows you to restart it where it stopped.
- Interrupt Terminates the calculation and returns you to the normal control panel.
- Computation Speed Allows computations to be performed at a range of speeds, from very slowly to as fast as the system will allow.
- Step Steps through the edge updates.

Figure 1.3 Control panel insert visible while triangulation is calculated



View Selections:

These selections appear on the permanent control panel (see Figure 1.2):

- Nice <-> Sloppy Allows the drawing to be sloppy (only local changes are updated) or nice (the whole screen is redrawn every time an edge is added, deleted, or flipped). Graphics are often a bottleneck, so Sloppy is much faster. Everything is completely updated for the final view of the triangulation; only the intermediate views are incompletely updated during a Sloppy operation.
- Spin Allows the user to rotate the image (left mouse button), zoom in and out (middle mouse button), and

return to the original orientation (right mouse button). If you press the left mouse button, move the mouse, and release the button while the mouse is still moving, the image will spin until you press the right button.

Autowireframe

Switches to wireframe mode while the object rotates.

Show Vertices

Shows only the vertices. All edges are hidden from view.

Show Lifted Vertices

Allows the user to examine the lifted points. (This is of interest only to those familiar with the lifting map.)

Weight (Z-coord) Off

Ignores the z-coordinate.

Shading

Several shading and lighting selections are also available in graphic mode:

- Vertices only
- Wireframe
- Lighting
- Hidden Line (version 1)
- Hidden Line (version 2)
- Minimum Angle
- Maximum Angle
- Minimum Height
- Maximum Slope

Shading is used to evaluate the quality of the triangulation. Light coloring generally indicates a better triangulation while darker coloring generally indicates a poorer triangulation.

Output Triangulations

After a triangulation has been calculated, the triangulation can be saved to a user-specifiable file. In non-graphic mode, the calculated triangulation is always automatically saved to a file named *algorithm_identifier.edges*.

Interrupt and Exit

You can interrupt algorithms without terminating MinMaxer only in graphic mode. If you wish to pause and plan to resume calculations shortly, select

PAUSE

If you wish to terminate the calculation, select

INTERRUPT

To quit MinMaxer in either graphic or non-graphic mode, select

QUIT

Special Notes

Graphic mode is rather slow for data sets larger than 5000 points.

Bugs and Cautions

While the plane sweep algorithm, by itself, produces only a marginally useful triangulation, users are encouraged to use it to initialize every data set to avoid difficulties with the other triangulations. For example, if the Lawson Flip algorithm is applied to a data set that has not been initialized, it returns an empty triangulation.

Chapter 2

Source Code Integration

Chapter Overview

This chapter provides a short guide for developers who wish to incorporate the MinMaxer triangulation functions into an application. No knowledge of the triangulation algorithms is required, though a person familiar with them will find the process more intuitive.

Resources

Source Code

The directories `source.minmaxer/` and `source.triangulations/` contain the MinMaxer source code for Sun and Silicon Graphics systems.

To compile MinMaxer for either a Sun or an SGI system, type

```
make
```

This source code should work with only minor modifications on any other computer. For example, a Cray system requires minor modifications to the timing functions. Many systems will not require any modification.

Sample Files

The directory `sample/` contains several sample programs and data sets that illustrate integrating MinMaxer with an application.

The names of the files are `testxxxx.c`, where `xxxx` names the triangulation employed by the file. Use the command

```
make testxxxx
```

to compile these programs. The `Makefile` may have to be modified slightly depending on whether you are compiling on a Silicon Graphics system.

The provided `Makefile` should be taken as a guide to decide which files must be included for a specific triangulation algorithm. If it is not clear which triangulation algorithms will be used and if memory is not a concern, it may be easiest to include all the files stored in the variables `TRIANGULATIONMINIMUMOBJ` and `ALLTRIOBJ`.

Copyright Requirements

Though NCSA MinMaxer source code has been placed in the public domain, it is copyrighted. Anyone using MinMaxer source code in an application must adhere to the copyright restrictions described on the page following the title page.

Implementation

Data Formats

MinMaxer assumes that the data point coordinates are stored in three integer arrays: x , y , and z . (Not all of the triangulations use the z -coordinate. See Table 2.1.) If there is no z -coordinate, z should be a pointer to NULL. The first entry in these arrays is assumed to be at position 0.

Data Translation

The first step in computing a triangulation is to transfer the coordinates to the internal representation. This representation is held in a structure pointed to by a variable of type `char *`. The procedure which transfers the data is

```
copyCoordinatesToGraph (n, x, y, z, r, &g)
```

where

- n is an integer denoting the number of vertices.
- x , y , and z are of type `int *` as described above and describe the coordinates of the data points. (If z is NULL, the z -coordinates in the internal representation will be set to 0.)
- r is an integer with a value of either 0 or 1. If r is 0, duplicates are not eliminated; if r is 1, duplicates should be eliminated. Two points are considered duplicates if both the x - and y -coordinates are identical (the z -coordinate might be different).
- g is of type `char *` and is a pointer to the internal data structure that MinMaxer uses to compute the triangulation..

Data Manipulation

The next step is to compute a triangulation. The triangulation functions are listed in Table 1, "Algorithms and Functions." Use the sample files in the directory `sample/` as templates for this calculation.

Table 2.1 names the triangulations and algorithms and identifies the function that performs each. The table also identifies the algorithms that require an initial triangulation, the algorithms that may fail if duplicate points are not removed, and the algorithms that use the z -coordinate.

Table 2.1 Algorithms and functions

	Function	Triangulation	Algorithm
*+	planeSweep	plane sweep	plane sweep
	delaunay1	Delaunay	Lawson flip
	delaunay2	Delaunay	incremental flip
\$	regular	regular	incremental flip
*+	minmaxAngle	minmax angle	edge insertion
*+\$	minmaxSlope	minmax slope	edge insertion
*+	maxminHeight	maxmin height	edge insertion

* Algorithm can be applied only to a pre-existing triangulation.

+ Algorithm may fail if duplicate points are not removed.

\$ Algorithm uses z -coordinate.

Triangulations are stored in a quadedge data structure. The functions that operate on the quadedge data structure can be found in `source.triangulation/quadedge.h`.

For those not familiar with the quadedge data structure, a conversion function which stores the triangulation as a list of triangles (i.e. triples of indices, where the index i corresponds to the vertex composed of the coordinates stored in i -th position of x, y , and z .) is provided. The function

```
copyGraphToListOfTriangles (g, &list)
```

where

g is the pointer to the internal representation.

list, of type `triangleList *`, is a pointer to the list of triangles.

The file `source.triangulation/triangulation.h` defines the type `triangleList`.

Data Storage

A triangulation can also be stored to a file. The file format is the same as the MinMaxer output format, a list of quintuples describing the vertices

```
site x-coord y-coord z-coord index
```

followed by a list of triples describing the edges of the triangulation

```
edge from-vertex to-vertex
```

The function

```
saveTriangulation (g, fileName)
```

dumps the triangulation stored in *g* into the file *fileName*.

A

ppendix

Installation and Online Structure

This appendix provides instructions for the following activities:

- Acquisition of the latest version of MinMaxer
- MinMaxer installation
- The MinMaxer directory and file structure

Acquiring the Latest Version

NCSA MinMaxer is available on NCSA's anonymous FTP server in the directory `/SGI/MinMaxer/`. The file `README.FIRST` in the root directory of the server contains a complete discussion of the following installation procedure. Read it if you have any questions that are not answered here.

If you have an Internet link, either of the following `ftp` commands will get you to the server:

```
ftp ftp.ncsa.uiuc.edu
or
ftp 141.142.20.50
```

Log in as `anonymous` using your email address as a password.

Once you are on the server, go to the MinMaxer directory and pick up the tarred and compressed MinMaxer file:

```
cd SGI
cd MinMaxer
get minmaxer.tar.Z
quit
```

Uncompress the file `minmaxer.tar.Z` on your system and extract MinMaxer:

```
uncompress minmaxer.tar.Z
tar xf minmaxer.tar
```

The MinMaxer files will be extracted and written to your system under the directory `minmaxer.version`. See the section "Directory and File Structure" below for a complete listing of the installed files.

No ftp connection?

If you do not have an Internet connection and cannot acquire MinMaxer via `ftp`, it is on the FTP Source Tape available from the Software Development Group at NCSA. This tape contains the entire contents of our FTP server. Our address is listed in the "Orders" section immediately following the title page of this document. There is a fee for this service.

Installation

After completing the steps above, move to the directory `minmaxer.version/minmaxer.bin`. You will see two executable files:

```
minmaxer.sgi
minmaxer.sun
```

If you will be working on a Sun system, copy `minmaxer.sun` to `MinMaxer`. If you will be working on an SGI system, copy `minmaxer.sgi` to `MinMaxer`. If you will be working on any other system, you will have to compile `MinMaxer` for that system.

Directory and File Structure

The `MinMaxer` files, installed as described above, are stored in the following directories:

<code>minmaxer.bin</code>	Executables for SUN-SPARC and SGI systems with sample input data
<code>sample</code>	Sample programs that illustrate the integration of <code>MinMaxer</code> and an application
<code>source.minmaxer</code>	<code>MinMaxer</code> source files
<code>source.triangulation</code>	Triangulation algorithm source files

A complete directory listing appears as Figure A-1.

Figure A.1 MinMaxer Version 1.04 installed directory structure and files

```

prompt_79% pwd
/your_dir/MinMaxer/minmaxer.v1.04

prompt_80% ls -F
COPYRIGHT          minmaxer.bin/      source.minmaxer/
README            sample/            source.triangulation/

prompt_81% ls *
COPYRIGHT          README

minmaxer.bin:
200.data           initial.edges      minmaxer.sun
README            maxmin.height.edges nice.regular.data
delaunay.edges    minmax.angle.edges sample.data
delaunay.verify   minmaxer.sgi

sample:
Makefile          testall.c          testdelaunay1.c  testheight.c     testslope.c
README           testangle.c        testfile         testregular.c

source.minmaxer:
Makefile          sgimenu.c          sgivisual.c      texput.c          vect.c
README           sgirender.c        sunvisual.c      theMakefile       vect.h
minmaxer.c       sgivislib.c        support.c         trackball.c
sgiformsmenu.c  sgivislib.h        support.h         trackball.h

source.triangulation:
README           hdag.h             quadedge.h
angle.c          heap.c             queue.c
angle.sos.c      heap.h             queue.h
bitvector.c      height.c           quicksort.c
bitvector.h      heuristic.angle.c  regular.c
delaunay.c       heuristic.height.c road.h
edgeinsert.c     heuristic.slope.c  slope.c
file_io.c        longmath.c         sos.c
file_io.h        longmath.h         stack.c
flips.c          menu.h             stack.h
flips.h          novisual.c         timer.c
geometry.objects.h persistent.quadedge.c triangulation.c
graph.c          persistent.quadedge.h triangulation.h
graph.h          planesweep.c       triation.h
hdag.c           quadedge.c         visual.h
prompt_82%

```

R

References

This section lists references that provide complete technical discussions of the triangulation algorithms used in MinMaxer.

M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T. S. Tan, "Edge insertion for optimal triangulations," *Proceedings of the First Latin American Symposium on Theoretical Informatics* (1992), pp 46-60. Also to appear in *Discrete and Computational Geometry*.

B. Delaunay, "Sur la sphere vide," *Izvestia Akademia Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7 (1934), pp 793-800.

H. Edelsbrunner, *Algorithms in Combinatorial Geometry* (Springer-Verlag, 1987).

H. Edelsbrunner and E. P. Mücke, "Simulation of Simplicity: a technique to cope with degenerate cases in geometric algorithms," *ACM Transactions on Graphics* 9, No. 1 (January 1990), pp 66-104.

H. Edelsbrunner and N. Shah, "Incremental topological flipping works for regular triangulations," *Proceedings of the 8th Annual Symposium on Computational Geometry* (1992), pp 43-52.

H. Edelsbrunner, T. S. Tan, and R. Waupotitsch, "An $O(n^2 \log n)$ time algorithm for the minmax angle triangulation," *SIAM Journal on Scientific and Statistical Computing* 13 (1992), pp 994-1008.

L. Guibas, D. Knuth, and M. Sharir, "Randomized incremental construction of Delaunay and Voronoi diagrams," *Lecture Notes in Computer Science* 443 (Springer-Verlag, 1990), pp 414-431.