APPENDIX B

PRISM-MAP IMPLEMENTATION

This appendix describes an implementation of the prism-map special case algorithm described in chapter 4. It contains a summary and portfolio, users' guide, and logic manual.
PRISM - A prism-plotting program

Summary & Portfolio

Wm. Randolph Franklin

Lab for Computer Graphics and Spatial Analysis

Gund Hall 520
Graduate School of Design
Harvard University
Cambridge, MA, 02138

Copyright, 1978.

March 23, 1978
Table of Contents

1. Introduction

2. Input Format

3. User Options
   General
   Shading
   Annotation
   Setting the Options

4. Implementation
   Method & Timing
   Generalizing the Input Map
   The Code
   The Plotter

5. Documentation

6. Availability
7. Portfolio
PRISM Summary & Portfolio

INTRODUCTION

1.0 INTRODUCTION

PRISM is an interactive program that the Lab for Computer Graphics offers that draws 3-D prism maps from 2-D maps. Although the corresponding 2-D map can be anything, consider one outlining regions such as the coterminous states. PRISM, operating in 3-D, erects on each state a prism whose height depends on some property of that state, such as population. Then it plots the result in 2 point perspective with the hidden edges deleted and the visible faces shaded.

2.0 INPUT FORMAT

The input map is in standard Odyssey GBF format. Either binary or formatted files are acceptable. The values file that gives the height of each prism is a formatted ASCII file with 1 line per polygon giving the polygon's number and value. This file can be prepared in a few minutes with a text editor. There is a standard collection of Odyssey GBF's giving the outlines of the states, counties, standard metropolitan statistical areas, etc. and more are being prepared. Thus if your field of interest is
one of these areas, your only data preparation is the values file.

The Odyssey GBF's are machine transportable in formatted form since they are then in ASCII card image format although on any given installation they are smaller and more efficient to use in binary form.

3.0 USER OPTIONS

3.1 General

PRISM has many user options: First you can choose your viewpoint, or the angle and distance of the imaginary 3-D prism model from the plotter screen. The viewpoint comprises the compass direction you are looking at the map from, your angle of elevation above the horizontal and your relative distance, scaled so that the diameter of the map is 2. If you are close, perspective effects appear; the farther parts of the map are shrunk.
3.2 Shading

PRISM's shading is controlled by 4 user parameters. The sides of the prisms are shaded by vertical hatch lines to approximate a light shining on them. The angle the light is coming from is variable as are the minimum and maximum spacing of the crosshatch lines. Another parameter controls the reflection law used so you are not restricted to a cosine law. Instead of vertical shading, horizontal contour lines can be drawn on the sides of the prisms. The horizontal top and bottom edges of the prisms may be deleted. The vertical prism edges rising from the points in the original map may be deleted.

3.3 Annotation

A box around the plot and a title at the bottom may be drawn. This title has the title of the map file, the title of the values file, a user supplied comment line and the date & time and user making the plot. The location of the plot on the screen can be changed.
3.4 Setting The Options

The parameters are set with a free format interactive language that allows you to set and display variables. However most of the options mentioned above can be ignored if desired since then reasonable defaults are used.

4.0 IMPLEMENTATION

4.1 Method & Timing

PRISM has 2 steps: The first is to read the map file and transform it given the viewpoint and sort it. A sorted file is written that can be read later to save the sorting step if you reuse the same map and viewpoint. The second step is to read the sorted file and the values file and draw the plot. This step runs fast enough to keep a 9600 baud CRT drawing constantly. You can change the values file, the titles, the maximum prism height and all the shading parameters without changing the sorted file. Thus you can quickly experiment with different sets of heights and types of shading without using much computer time. The first step for a typical 1000 point map takes about 20 seconds. This is on a PDP KA-10. A very large map with 4641 points takes
88 seconds. While a map with only 367 points takes 7 seconds.

4.2 Generalizing The Input Maps

PRISM is enhanced by Odyssey's Elf, a program written by Dennis White that generalizes or reduces map files to different levels of accuracy. This is useful because the highly generalized maps produce blocky prism maps that most people find more appealing. The smaller more generalized maps are also quicker for PRISM to sort.

4.3 The Code

PRISM is a 6000 line program written in FLECS, a Fortran preprocessor developed by Terry Beyer at U. of Oregon. FLECS adds block structure and internal routines to Fortran. The output is a standard Fortran program and this is what is distributed.

PRISM has been implemented on the DEC PDP-10 and IBM-370/TSO. Since it is distributed in standard Fortran it can be easily converted to other machines. For efficiency,
some small routines are also coded in machine language on the PDP-10 although Fortran equivalents are available for speed of conversion.

PRISM uses overlaying and on the PDP-10 takes 21+7K when compiled with Fortran-10.

4.4 The Plotter

PRISM plots on a Tektronix 4010 or 4014 CRT's or Calcomp Plotters. Moreover all plotter calls interface through small routines that can be easily updated to handle other plotters. The only primitive plotter commands used are to move the pen and draw a line; even the optional titles have letters generated from a table.

5.0 DOCUMENTATION

PRISM is well documented with a user manual, a program logic manual and an algorithm description. The user manual summarizes its capabilities, gives the commands and tells how to use it on PDP-10 and IBM-TPS. It also gives the distribution tape format. The logic manual describes in
detail the routines, common variables, data file formats, I/O units, library routines, overlay structure, calling tree, etc. The algorithm description concentrates on the method PRISM uses as distinguished from the actual implementation. Included with this summary is a portfolio showing some sample prism plots illustrating different capabilities.

6.0 AVAILABILITY

PRISM is available for license on a single CPU within a single organization from the Lab for Computer Graphics for a $1000 initial fee plus $100 per year to educational and governmental institutions or for $1500 and $150 to commercial organizations.

PRISM is distributed on a 9 track unlabelled tape written at 800 BPI containing several files with the source, the users' guide, the program logic manual, and sample files.
7.0 PORTFOLIO

Here follow some sample plots that illustrate different aspects of PRISM such as the shading, rotation of the viewpoint, different values files, etc.

1. Base map of the conterminous USA with 1228 points.

2. PRISM-map of the above where the height of each prism is proportional the the estimated number of alcoholics per 100,000. The shading is assuming an imaginary light source from the left.

3. Like Figure 2 but looking from the north.

4. Again alcoholism but looking from the north-east, from close enough that there is noticeable foreshortening and using a base map that is a rougher approximation and has fewer points.

5. Another PRISM-map from Figure 1, this time on public school expenditures, dollars per capita.

6. Another map, this time on percent illiteracy. By comparing Figures 5 and 6 you can quickly see that some states such as New York have both high per capita school
expenditures and high per capita illiteracy.

7. A map showing percentage of school children riding a bus to school.

8. The same as Figure 7 but without the shading.

9. Census tracts in Fresno, Calif. showing percentage of high school grads.

10. The same as Figure 9 but drawing horizontal contour lines every 5% on the sides of the prisms instead of shading.

11. A base map showing SAMI regions in the USA. SAMI regions are used for marketing surveys and forecasts.

12. A PRISM-map of the SAMI regions with random heights.

13. A PRISM-map produced from input created by PREPRS a preprocessor written by John Quarterman that assumes the prisms are buildings and provides suitable operations to manipulate them.
ALCOHOLISM
PUBLIC SCHOOL EXPENDITURES
dollars per capita.

Figure 5
ILLITERACY
per cent

Figure 6
percentage of SCHOOL CHILDREN BUSSED

Figure 7
Fresno percentage of high school grads
FRESNO percentage of high school grads

Figure 10
PRISM

A Prism Map Plotting Program

Users' Manual


Wm. Randolph Franklin

Laboratory for Computer Graphics and Spatial Analysis
Gund Hall 520
Graduate School of Design
Harvard University
Cambridge, Mass., 02138

(617) 495-2526

ARPANet: FRANKLIN@HARV-10

Copyright, 1976, 1977, by the President and Fellows of Harvard College, Cambridge, Mass., 02138. All rights reserved.
1.0 TABLE OF CONTENTS

   Chapter
   ------
1. Table of Contents
2. Introduction
3. Data Files
4. How to Specify the Files
   PDP-10
   IBM-370
5. The Plotter
6. Shading in the Plot
7. Other Plotter Options
8. Running PRISM
9. Using Other Computers
10. Distribution Tape Format
11. Examples
2.0 INTRODUCTION

PRISM is an interactive program to draw prism maps. For example, consider a map such as Figure 1 outlining regions such as the coterminous states. This map is in 2-D. PRISM, operating in 3-D, erected on each state a prism whose height depends on some property of that state, such as population. Then it plots the result in 2 point perspective with the hidden edges deleted and the visible faces shaded (see Figure 2).

You can rotate around your prism map & change your elevation, as defined by the VIEWPOINT which is the imaginary point from which you are looking at the plot. The viewpoint is defined by 3 variables:

1. VPANGH: Your horizontal angle of rotation relative to the scene, in degrees, anticlockwise. 0 means looking from (0, -infinity).

2. VPANGV: Your vertical angle above the horizon, in degrees.

3. VPDIST: Your relative distance from the centre of the map along the line determined by the first 2 angles. The centre is the middle of the box around the map given in the data file. This box is rotated about its centre by VPANGH and a new box orthogonal to the axes is constructed bounding this new oblique box. Now the scale of VPDIST is such that 1 is the distance from the centre to any corner of this new box. If you specify VPDIST < 0, you get an inverted perspective where closer parts of the map are shrunk relative to the more distance parts. (Normal perspective is the reverse). You must have ABS(VPDIST) big enough so that the viewpoint can be separated from the map by a vertical plane; interior views are not allowed. ABS(VPDIST) > 1 is certainly OK, but somewhat smaller values may be legal also, depending on the map's shape & angle of rotation.

A typical viewpoint is (0., 20., 10000.).
3.0 DATA FILES

Basically what PRISM does is read a 2-D map, rotate, project and scale it, the read and plot heights files for that map.

The input map is a chain file & associated global file in the Odyssey format. PRISM rotates and projects the map according to the viewpoint and then splits the chains up into edge segments and sorts them. Next it writes the sorted edges in a file that can be used in the future instead of the original map file so long as you keep the same viewpoint. This is useful since this sorting is by far the slowest part of PRISM. You can define the prism heights in a separate file formatted thus: The first line is a 20A4 description of the file that is made part of the plot title. Thereafter the format is (I10,2XA4,4X,F10.3) with 1 line per polygon containing the polygon identifier, a short title & its height. Follow the last line with '9999' (that is six blanks followed by four nines). The polygons may be listed in any order. The file is read until the 9999 and heights for any polygons that are not in the chain file cause a message and then are ignored. Thus you can use a heights file that was prepared for an area with more polygons even though you are only mapping part of it now. Prisms that have no heights in the file are assigned heights of 0. The polygon heights are matched with the chains by the polygon identifiers; the titles are used only to label the plot.

If you have several sets of heights, put each set in a separate file. You can run several sets with the same input chain or sorted file since the heights don't affect the sorting.
4.0 HOW TO SPECIFY THE FILES

4.1 PDP-10

PRISM will prompt for the names which must be 1-5 chars with extension DAT. This restriction will be removed with the Fortran-10 compiler. If you don't specify an output sorted file name, a random name will be chosen. If you give a name, it must be new for PRISM never overwrites an existing file.

PRISM also uses 7 temporary files named 0XXXN.TMP where XXX is 3 random letters and N a digit. If PRISM ends normally it deletes them; else they will remain for you to delete.

4.2 IBM-370/TSO

The file (*) names are defined by the following TSO commands, possibly in a user command file:

Case 1: reading map from chain & global files:

FREE FILE(SYSPLIC, SYSPROC, SYSIN)
'ATTRIB A1 RECFM(V B S) LRECL(512) BLKSIZE(2052)
ALLOC FILE(PT01F001) DA(PRERR.DAT) OLD
ALLOC FILE(PT02F001) DA(US.GLOBAL) OLD
ALLOC FILE(PT03F001) DA(TMP.P3) NEW BLOCK(2052) -
   SPACE(20 20) USING(A1)
ALLOC FILE(PT04F001) DA(US.EDGE) NEW BLOCK(2052) -

(*) Called a DATASET by IBM. What IBM calls a file is synonymous with a dataset for these purposes.
PRISM - A Prism Map Plotting Program

HOW TO SPECIFY THE FILES

SPACE(20 20) USING(A1)
ALLOC FILE(FT07F001) DA(TMP.P7) NEW BLOCK(2052) -
SPACE(20 20) USING(A1)
ALLOC FILE(FT08F001) DA(TMP.P8) NEW BLOCK(2052) -
SPACE(20 20) USING(A1)
ALLOC FILE(FT09F001) DA(TMP.P9) NEW BLOCK(2052) -
SPACE(20 20) USING(A1)
ALLOC FILE(FT10F001) DA(TMP.P10) NEW BLOCK(2052) -
SPACE(20 20) USING(A1)
ALLOC FILE(FT11F001) DA(TMP.P11) NEW BLOCK(2052) -
SPACE(20 20) USING(A1)
ALLOC FILE(FT12F001) DA(TMP.P12) NEW BLOCK(2052) -
SPACE(20 20) USING(A1)
ALLOC FILE(FT13F001) DA(US.CHAIN) OLD
ALLOC FILE(FT14F001) DA(US.HEIGRTS) OLD

Replace US.CHAIN, US.GLOBAL and US.HEIGRTS by your own files. The output sorted file is US.EDGE. If you just want a sorted file and not a plot you don’t need to allocate the heights file.

Case 2: Reading map from sorted edge file:

ALLOC FILE(FT04F001) DA(US.EDGE) OLD
ALLOC FILE(FT14F001) DA(US.HEIGRTS) OLD

5.0 THE PLOTTER

PRISM assumes you running it interactively on a Tektronix 4010 or 4014 graphics terminal. The only special feature of the 4014 that is used is the 12 bit addressing that makes lines on the plot meet slightly (but noticeably) more accurately. To the extent that 4002 routines are compatible, it can also be used with no changes to PRISM.

To facilitate using other plotters, all plotter calls are made through the following fundamental machine independent routines: PBOBS (set bounds of plot), POPEN (initialize plotter), PCLEAR (clear screen), PCLOSE (finish
plotting), PMOVE (move pen) and PDRAW (draw line). Since they are all so primitive, they can be redefined in terms of any other plotter's calls easily.

The plot screen is assumed to be 4096 by 3120 on the PDP-10 and 1024 by 780 on the IBM-370.

6.0 SHADING IN THE PLOT

PRISM has several options to allow you to choose the most effective shading of your plot. Only the sides of the prisms can be shaded, however. To use the shading fully, you must know how PRISM does it, although leaving all the options at their default values generally works.

You can shade the sides of the prisms by setting variable KSHADE. -1 means don't. Else vertical lines are drawn on each visible side of each prism at a spacing dependent on the angle between the normal to the side and the assumed incident light. Assume light is shining from infinity at an angle LITANG (counterclockwise from the viewpoint, in degrees). Call the angle between LITANG and the normal to the face R1. -180 < R1 <= 180. However for any given LITANG< R1 is usually more restricted than this because faces inclined so as to cause the extreme angles are faced away from the viewpoint and thus invisible and not shadable. For instance if LITANG = 45, visible faces will have -45 <= R1 <= 135.

Let R2 = ABS(R1), normalized to run< from 0 for the lightest possible visible face to 1 for the darkest. The reason for this normalization, dependent on LITANG, is to keep the average density of the shading more nearly constant as you change LITANG. This density will still change somewhat since the statistical distribution of the faces relative to the light changes in a manner dependent on the individual plot as LITANG changes.
Next R2 is transformed to \( R3 = F(R2) \) using the probability distribution function \( F \). The purpose of \( F \), which has user-settable parameters is to let you experiment with different shading laws. \( F(x) \) is derived from \( f(x) \) thus:

\[
F(X) = C \cdot \text{Integral (from } 0 \text{ to } X \text{ of } f(x) \)
\]

where \( f(x) = x^a\cdot(1-x)^b \) and \( C \) is a normalizing constant. Thus \( 0 \leq R3 \leq 1 \). \( A \) & \( B \) are the digits of KSHADE, which if not -1 is considered as the 2 digit integer \( AB \). (An implicit leading \( 0 \) is added if necessary).

Finally \( R3 \) is changed to \( R4 \) by a linear transformation so that \( R4 \) ranges from LOSPAC for the darkest shading to HISPAC for the lightest. Then vertical shading lines are drawn every \( R4 \) plotter units. If you don’t want variable spacing, set LOSPAC and HISPAC the same.

Thus you can control the shading with 5 parameters: LITANG, A, B, LOSPAC & HISPAC.

Effects of different A & B: \( A=B=0 \) (KSHADE=0) means \( R3=R2 \) so the density of shading depends linearly on the angle of incidence of the light on the face. \( A=1, B=0 \) (KSHADE=10) gives \( R3 = R2^2 \) which approximates a cosine law. As \( A \) increases over \( B \), the average shading gets lighter. Conversely, if \( B > A \), it gets darker. If \( A=B=1 \), the plot has the same average darkness, but differences in the extreme angles of incidences are deaccentuated and differences in medium angles made more contrasting. The effect is even more pronounced if KSHADE=22. This is similar to using a high contrast film.

The legal values of KSHADE are \(-1, 0, 1, 2, 10, 11, 12, 20, 21 \& 22 \). See figure 3 for the nonlinear effects of the different values.

The reason for having a variable shading function instead of just plugging in the physically correct shading law is that the shading is to make the plot easier to understand and quickly grasp, not to mimic what a cardboard model would look like if illuminated by a spotlight.

The current shading algorithm treats each edge of a chain (and thus each face of a prism) separately. If the edges are smaller than shading spacing, this causes
problems. A random number generator keeps the average density correct, but the lack of coordination between adjacent faces causes blotchiness. This can be reduced by using Odyssey's ELF to reduce the chain file to about 1000 edges *. If the edges are longer than the shading spacing, the shading runs smoothly and invisibly across face boundaries.

* ELF is a program written by Dennis White a LCGSA whose relevent functions are:
1. Plot a chain file on the screen.
2. Generalize a chain file to a given accuracy. This means that each chain is replaced by a chain with fewer points that is always within epsilon of the original chain for some given epsilon. The resulting file uses less space, is faster to plot and looks better when shaded. The resulting prism maps have a blocky look that is sometimes appealing.
PRISM - A Prism Map Plotting Program

OTHER PLOTTER OPTIONS

7.0 OTHER PLOTTER OPTIONS

You have the following choices for the plot:

The maximum height of the prisms is settable as a fraction of the height of the plot. You can control whether or not to draw titles at the bottom of the plot and a box around it. You can shade the sides of the prisms in vertical lines.

You can also draw horizontal contour lines at given intervals. That is the contour lines are horizontal in the original - on the plot they are generally oblique. You can give the contour interval as either a certain distance in the units of the heights (by setting CONHTS) or as a certain distance on the plot (by setting CONPLO). Which method is used depends on which variable is nonzero. If both are nonzero, user units are used. If both are zero, default plotter units are used. The horizontal and vertical edges of the prisms can be omitted since with the sides' shading they are not necessary. You can also just draw the 'silhouette' vertical edges; those defining the boundary between a prism and another prism or background. The numbers of the polygons can be printed on the prisms. You can cause all the prisms' heights to be shifted so that the lowest is 0. This is useful if there are negative heights (normally illegal) or all the heights are approximately the same & you wish to accentuate the differences. Finally you can list a help text giving one line definitions of the variables and their current values. Here are the variables and initial values: Whenever plot units are mentioned, the screen is 1024 horizontal by 780 vertical.

Summary of variables & initial values:

- CONHTS: Spacing of horizontal contour lines in user units (0).
- CONPLO: Spacing of the horizontal contour lines, in plotter units. (8).
- HELP: If you assign a value to this variable, one line descriptions and current values of all the variables are listed.
- HISPAC: The greatest spacing of vertical shading lines in plotter units. (30)
PRISM - A Prism Map Plotting Program

OTHER PLOTTER OPTIONS

KSHADE The type of vertical shading on the prism sides:
(0)
-1 None.
0 Standard shading with the density depending on the lighting.
>0 As above but with nonlinear shading laws.

LABEL Print the polygon titles on the prism tops? (FALSE)

LCONTR Should horizontal contour lines be drawn? (FALSE).

LDRBOX Should a box be drawn around the plot? (TRUE).

LDRHOR Should the horizontal edges of the prisms be drawn? (TRUE)

LDRSIL If the vertical edges are drawn, should only the silhouette vertical edges be drawn? (TRUE).

LDRTIT Should a title be drawn at the bottom of the plot? (TRUE).

LRVRVER Should the vertical edges of the prisms be drawn? This should generally be FALSE for shading and TRUE without. (FALSE).

LITANG The angle of illumination for vertical shading, in degrees, positive rotation from your viewpoint. (-90.0).

LLOHTZ Should the heights be scaled so that the lowest one is zero? (FALSE).

LOSPAC Closest spacing of vertical shading lines, plotter units. (3).

LTIME Should time & frequency counts be kept for each routine? This makes PRISM run twice as slow. (FALSE).

PLZPCT The max prism height as a percentage of the height of the screen. (30.0).

8.0 RUNNING PRISM

On the IBM-370 with TSO, you need your input files and the PRISM load module. Allocate the files as shown above then type 'CALL PRISM' to execute it, answering the questions, if they are asked, as for the PDP-10. You are not asked for any file names since they are already defined.
on the DD cards. On IBM you can only have one heights file. However you can plot it several times with different plot parameters. To use a different heights file, allocate the new one and run PRISM again.

On the PDP-10, you need the program, PRISM.SAV & the error message file, PRERR.DAT (*). You also need your data files. Copy these files into your account, then type RUN PRISM to start.

The first question is whether you want the map data read from a chain file or from a sorted edge file. Type 1 or 2 respectively. If you type 2 then the next 4 questions are bypassed. To the next 2 questions, type the chain & global files names. Then PRISM asks for the name of the sorted edge file it will write. If you type just a carriage return, PRISM assigns a random name and tells you.

The next question is for the coordinates of the viewpoint (3 reals): it is read in (3E15.5) format and commas are printed on the line above to help you align it.

Now PRISM asks whether you want to read a heights file and plot it, plot using random heights or end. Type 1, 2 or 3 respectively. If 1 then PRISM asks for the name of a heights file. Random heights are useful if you have some map data but no prepared heights file & you just want to see what a typical prism map plot from it would look like.

After the heights file PRISM gives you the chance to change the plot parameters in a Namelist statement called IN. Type

```
<space>$IN name=value name=value ... $end<return>
```

You can continue onto multiple lines between names, but leave the first space of every line blank. If you don't get the namelist format right, PRISM will just sit there waiting for a correct namelist block. Once you type in a correct

--------------------

(*) Only needed if PRISM has a fatal error when it allows a line of description to be printed instead of just an error number.
PRISM - A Prism Map Plotting Program
RUNNING PRISM

block, PRISM will type something almost immediately. If you assign the variable HELP, after the variables are listed, PRISM prompts for the namelist block again to give you a chance to change more variables. Misspelling a variable name, at least on the PDP-10 F40, is a terminal error.

On IBM-370, use & (ampersand) instead of $ (dollar sign) for the namelist.

Now that everything has been specified, when PRISM types 'READY TO PLOT', type a carriage return. Type another when you are finished looking at the plot. Then you are again asked whether you want to read a heights file, use random heights or end. As before, type 1, 2 or 3. If 3 then PRISM deletes its temporary files and stops. At Harvard it also prints the job's cost. If you are reading a chain file and not a sorted file you might want to end without plotting anything (but after the sorted file has been written).

To use a different map or viewpoint, just run PRISM again.

9.0 USING OTHER COMPUTERS

PRISM is a 4000 lines FLECS program. FLECS is a Fortran preprocessor adding block structure and in-line routines obtainable from

Terry Beyer,
Computing Center,
University of Oregon,
Eugene, Oregon, 97405.
(503) 686-4416

FLECS is written in FLECS but has some machine dependent routines for speed. FLECS produces output violating the Ansi standard in that it has do loops with extended ranges that themselves contain do loops. This has not yet been a problem.

Prism has some machine language routines that fall into 4 classes:
PRISM - A Prism Map Plotting Program

USING OTHER COMPUTERS

1. Operations that can be done faster than in Fortran such as combining 2 halfwords into one fullword. Fortran equivalents exist for these routines.

2. Pretty printing functions to print the user name, date and time on the plot and the job cost when you end PRISM. They are controlled by common variable MACHIN which gives the machine type. If it is not PDP-10, none of these are called. There is a file of dummy routines for these to satisfy external references. The job cost is only printed if common variable INTERN(2).GE.1 since it is Harvard specific.

3. Plotter routines are described above.

4. Various debugging routines such as machine dependent plotter calls are used depending on the settings in array INTERN and logical LT Timer. None are called with the defaults. PRISM is laced with debugging aids such as routine tracers and timers, write statements etc.

   If you have questions, comments or find bugs, phone or write the Lab for Computer Graphics. I welcome comments such as what you found difficult or counterintuitive or what should be included.

   There is a PRISM Program Logic Manual which is a 40 page document describing the program routine by routine.

   On the PDP-10 with FORTRAN-40, PRISM takes 32+5K. On a decent compiler it would be smaller & probably faster.

10.0 DISTRIBUTION TAPE FORMAT
PRISM - A Prism Map Plotting Program

DISTRIBUTION TAPE FORMAT

10.1 IBM-370

The tape is 200 foot unlabelled 800 BPI. DCB=(LRECL=80,BLKSIZ=800,RECFM=FB). The files are:

1. PRISM Fortran source code.
2. The file of PRISM error messages, PRERR.DAT.
3. The PRISM users' guide in upper and lower case text with carriage control chars in column 1.
4. The PRISM Program Logic Manual in similar format.
5. A sample global data file for the US state outlines.
6. The corresponding formatted chain file.
7. A corresponding heights data file.
PRISM

A Prism Map Plotting Program

Program Logic Manual

Wm. Randolph Franklin

Laboratory for Computer Graphics and Spatial Analysis
Gund Hall 520
Graduate School of Design
Harvard University
Cambridge, Mass., 02138

(617) 495-2526

date: 24-oct-77 10:44:26

Copyright, 1976, 1977, by the President and Fellows of Harvard College, Cambridge, Mass., 02138. All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage or retrieval system, without permission in writing.
1.0  TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Routine Names</td>
<td>3</td>
</tr>
<tr>
<td>Detailed Routine Descriptions</td>
<td>4</td>
</tr>
<tr>
<td>Calling Tree</td>
<td>24</td>
</tr>
<tr>
<td>Machine Dependencies</td>
<td>25</td>
</tr>
<tr>
<td>Common Blocks</td>
<td>26</td>
</tr>
<tr>
<td>Common Variables in PCOM</td>
<td>27</td>
</tr>
<tr>
<td>Data File Formats</td>
<td>30</td>
</tr>
<tr>
<td>Global Chain File</td>
<td></td>
</tr>
<tr>
<td>Chain File</td>
<td></td>
</tr>
<tr>
<td>Heights File(s)</td>
<td></td>
</tr>
<tr>
<td>Edge File # 1 (partly sorted)</td>
<td></td>
</tr>
<tr>
<td>Edge File # 2 (sorted)</td>
<td></td>
</tr>
<tr>
<td>Error Message File</td>
<td></td>
</tr>
<tr>
<td>Units Used</td>
<td>33</td>
</tr>
<tr>
<td>Library Routines Used</td>
<td>34</td>
</tr>
<tr>
<td>Using SOCKS Routines</td>
<td>34</td>
</tr>
<tr>
<td>Simple Statistical Analysis</td>
<td>36</td>
</tr>
<tr>
<td>Things to Do &amp; Thoughts for the Future</td>
<td>37</td>
</tr>
<tr>
<td>Disk Files</td>
<td>37</td>
</tr>
<tr>
<td>Including PRISM in another program</td>
<td>38</td>
</tr>
</tbody>
</table>
17. Overlay Structure
2.0 INTRODUCTION

This document describes the internal workings of the PRISM map program. It is not a users' manual - that is another document. This one describes every routine, common variable, I/O unit, data file and library routine used. It is written from a PDP-10 point of view since that is the primary system for PRISM. Included for completeness are several routines used by PRISM written by Dennis White & Nick Chrisman.

3.0 ROUTINE NAMES

This includes PRISM, SOCKS and plotting routines. It doesn't include LCG10 and LCG10B routines since they are adequately documented in their own manuals.

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK DATA</td>
</tr>
<tr>
<td>MAIN</td>
</tr>
<tr>
<td>COSTS (DAYTIM, RUNTIM, KCS, COST, IDAYTI, IRUNTI, IKCS, ICOST, RATE)</td>
</tr>
<tr>
<td>DAYTIM(DT)</td>
</tr>
<tr>
<td>ERRS (ROUTNAME, ERRNUM)</td>
</tr>
<tr>
<td>EXSORT(NREC, RECLN, INFILE, OUTFILE)</td>
</tr>
<tr>
<td>GVEC (CHAR, XX, YY, PEN, NV)</td>
</tr>
<tr>
<td>HASH (BASE, LEN, KEY, LOC, KODE)</td>
</tr>
<tr>
<td>HEAP0 (KHEAP, HEAPLEN/KREC, KEYLOC/0, KODE)</td>
</tr>
<tr>
<td>INVPRED(KK, N)</td>
</tr>
<tr>
<td>ITTOME (ISOCK, KNEE, LEG, EOF)</td>
</tr>
<tr>
<td>LINGO</td>
</tr>
<tr>
<td>LING02 (KODE, LOLD)</td>
</tr>
<tr>
<td>MEFILE (UNIT, FILESEQ, NAME, TYPE, LRECL) FN</td>
</tr>
<tr>
<td>MEFIL2 (UNITNO, FILENAME, KODE, ROUTNO, ERRNO)</td>
</tr>
<tr>
<td>NEWSOX (ISOCK, KODE, LEAD, EOF)</td>
</tr>
<tr>
<td>PBDS (XMIN, XMAX, YMIN, YMAX)</td>
</tr>
<tr>
<td>PCLEAR</td>
</tr>
<tr>
<td>PCLOSE</td>
</tr>
<tr>
<td>PDRAW (X, Y)</td>
</tr>
<tr>
<td>PLOTTR (X, Y, N)</td>
</tr>
<tr>
<td>PMOVE (X, Y)</td>
</tr>
<tr>
<td>PNUMB (X, Y, NUMB)</td>
</tr>
<tr>
<td>POPEN</td>
</tr>
</tbody>
</table>
The routine # above is a unique id for the routine in calls to TIME & ERRS.
4.0 DETAILED ROUTINE DESCRIPTIONS

4.1 MAIN

If PDP-10: Initialize the timer. Call MEFIL2 to randomize the temporary file names to PXXXI.DAT where XXX are 3 random letters and I is from 1 to 8. Set IOMSG, IOCMD to 5 so TEKBA5 won't bomb.

In any case: Prompt for the horizontal and vertical angles of the viewpoint. Prompt for the names of the chain file and its global file. Call READCH to read and reformat the chains as edges and then SRTEGD to sort them. Then repeatedly prompt for the name of a heights file and until it is blank, call PRPLOT to draw a plot.

Finally delete temporary files & print the job cost.

4.2 COSTS (DAYTIM, RUNTIM, KCS, COST, IDAYTI, IRUNTI, IKCS, ICOST, RATE)

Return some statistics on the job: the connect time (secs), cpu time since login (secs), core used since login (kilo-core-seconds), and cost of these ($). The first 3 are integers, the last real. Return integer RATE as A4, 'HI' or 'LO', depending on the time of day. If it is now noon - 8 P.M., Mon-Fri; everything since the last COSTS call is charged at hi rates. Else the lo rates are used. I don't check for holidays. Also the incremental stats since the last call to COSTS are given. The following hi rates are used. Lo rates are 1/3 of these:

$3.00 / connect hr.,
$0.006 / cpu sec.,
$0.0006 / kilo-core-sec.

There is one problem: Since I don't know how to get a job's connect time, the first time you call COSTS it is returned as 0 and the cost calculated from just the cpu time and core. Thereafter the connect time is calculated from that first call to COSTS.
If you don't want all these stats, the unwanted arguments may all be dummied with the same variable.

Note to people from other institutions wondering about our low rates: We have a very slow computer. A single precision floating divide takes 17 microseconds and there is no hardware double precision.

4.3 DAYTIM (DT)

Return the current date & time 5A4 in DT(5) formatted thus:

12-Jan-76 11:22:33
11112222333344445555

4.4 ERRS (ROUTNAME, ERRNUM)

ERRS is called from routine #IABS(ROUTNO). If ROUTNO > 0, it prints a terminal error message. In either case, it reads message file MSGFIL on unit #IOERR until the routine and error numbers match & prints that message. Then if ROUTNO > 0, it stops.

If the messages are put on a comment line immediately after the ERRS or MEFIL2 calls, the TECO macro ERRINC.TEC will extract them into file PRERR.DAT.

4.5 EXSORT (NREC, RECLLEN, INFILE, OUTFIL)

Externally sort a file of NREC records each of length RECLLEN using a 3 way merge and 6 temporary units. The input file, named INFILE is deleted. The output file is OUTFIL. ll of ISP after NISP is used as a workspace.

INFILE must be in buffers of length BLKSIZ with each containing as many records as possible. Records are not panned between buffers. Read the file in workspaces of length MISP - NISP words. SORT a workspace and write each un alternately to units 4-6. Define a run to be a list of
sorted records. Start reading files 4-6 and writing to 1-3. Merge the 3 first runs of 4-6 into one run 3 times larger and write to 1. Repeat writing to 1-3 in rotation. Then merge from 1-3 to 4-6 and back again until the whole file is 1 run. For each merge, only 1 buffer of each run being merged needs to be in core at a time. The first record of a run is a dummy whose first word is the number of records to follow in the run. When at the start of a merge pass, there is not more than 1 run in each input file, I know that this is the final pass so I 1) direct the output to OUTFIL, and 2) don't write the initial dummy record.

VARIABLES:

CURBUF current unit for writing.
IDISP(4) displacement in ISP to first free space in output buffer if I=4, or first unread record in the I-th input buffer, if I<=3.
INFILE(3) Name of input file.
NMR(3) The number of records left in each file's input buffer when merging.
NMW the number of records in the merge output buffer.
NWU The number of words of a buffer used for records.
NRECS(3) The number of records left in the current run in each read file.
NRUNS(6) the number of runs left to read on input or written on output.
OUTFIL(3) The name of the output file.
RECLen length of a record in words
RPB number of records per buffer
RPW number of records that fit in ISP
UNITS(6) the numbers of the temporary units

4.6 GVEC(CHAR, XX, YY, PEN, NV)

This routine converts the Ascii character code (1-127) in CHAR to a series of lines suitable for plotting. XX and YY store the points and PEN whether the pen is up (2) or down (3). NV is the number of points. XX, YY and PEN are integer arrays dimensioned (NV). XX and YY range from 0 to 6.
4.7 **HASH(BASE, LEN, KEY, LOC, KODE)**

Implement a hash table of size LEN words based at BASE. KEY is the nonzero integer key of the element. LOC is returned as its location in the table relative to the base. You can implement multiword hash tables by using LOC to index other arrays. You can have several simultaneous hash tables. They are distinguished by BASE.

**KODE OPERATION**

**ZHAINI** Initialize the table with BASE & LEN. Clear all entries. BASE and LEN must be unchanged on future calls to HASH.

**ZHALOC** Return the location of KEY or 0 if it doesn’t exist.

**ZHAADD** Add KEY to the table and return its location. Return 0 if it is already in the table.

**ZHALAD** Locate KEY if it exists or add it if it doesn’t. In any case return its location.

**ZHASTT** Print overflow stats: the number of times probes have overflowed I locations, 0 <= I <= 10.

*Method:* The first probe is at

\[
LOC = 1 + LEN \times \text{MOD} (\text{ABS}(KEY) \times 0.717483, 1)\]

0.717438 is just a random number. If LOC is occupied, successively increasing locations are tried until we are back at the start. This works fast enough as long as the table is under about 75% full. The array OVERFL(MOVERF) keeps count of overflow probes. It is zeroed when the hash table is init. Thereafter OVERFL(1) is the number of probes that did not overflow, OVERFL(2) is the number that overflowed 1 location, etc. and OVERFL(MOVERF) is the number that overflowed MOVERFL-1 or more locations. If there is more than 1 active table, these stats are for both tables since the last init.
4.8 HEAPØ (KHEAP, HEPLLEN/KREC, KEYLOC/Ø, KODE)

Implement a heap. To a heap you can add a record or return & delete the smallest record. The heap is in (ISP (KHEAP+I), I=1, HEPLLEN). The record is in (ISP (KREC+I), I=1,...). Records are only compared on 1 word: #KEYLOC; i.e. ISP(KREC+KEYLOC).

The first call is:

CALL HEAPØ(KHEAP, HEPLLEN, KEYLOC, ZHEINI)

The others are:

CALL HEAPØ(KHEAP, KREC, Ø, KODE)

After the initialization call, HEPLLEN, KEYLOC and NHEAP= the number of elements in the heap are stored in the first LHEDAT=3 locations so that the actual maximum number of entries is HEPLLEN - LHEDAT. The least element is ISP( KHEAP + LHEDAT + 1). You can have several active heaps at once.

KHEAP   (In) Base of heap.
HEPLLEN (In) The size of the heap in words & maximum number of records allowed in the heap.
KEYLOC  (In) The relative location in each record of the key for comparison.
KREC    (In/Out) The base of record to be added or being returned.
KODE    OPERATION
ZHEINI  Init the heap (required at start).
ZHEADD  Add REC to the heap.
ZHEDEL  Return smallest record & delete it.

In the heap, if A[I] is the base of the I-th record, its sons are in A[2*I] and A[2*I+1]. A record is less than both its sons, tested on the comparison word.
To add a record, insert it as the N+1-st element. Then work back to the root, exchanging it with its father if it is smaller and repeating on the father. To delete, the smallest record is #1. Replace it by record #NRECS. Then exchange it with the smaller of its sons if it is larger than that record and repeat on that son. The records themselves are never moved, only the pointers to them.

HIHEAP holds the maximum number of elements in any heap that has had an element added since the last heap initialization.

HEAP0, formerly called HEAP, was renamed to avoid conflict with a parser common block.

4.9 INVPER (KK, N)

Invert the permutation of N elements based at KK. This is called from SORT but is currently never used.

4.10 IOALO (I) Function

Allocate and free I/O units. I is not changed. No I/O is done. Only an allocation table is changed.

I= 0 Allocate and return a free unit number.
I=-1 Free all unit numbers except IOTTY & set their devices to DSK:.
I> 0 Free unit #I.

4.11 ITTOME(ISOCK, KNEE, LEG, EOF)

Return a socks chain read from channel ISOCK (1 to 3). The fixed part is in array KNEE and the variable part in LEG. Logical EOF is set if an en of file is read.
4.12 LINGO

This is the routine that will parse the user command language.

4.13 LINGO2(KODE, LOLD)

This checks the data returned from LINGO for consistency and validity. It keeps calling LINGO until the data is OK and then returns with KODE = ZLIEND or ZLIPLO telling PRISM to stop or plot. They are lexeme numbers in lingo and must be changed if they are changed in the language. LOLD is returned TRUE if SRTFIL is to be read. Else a GBF is to be read and STRFIL written. LOLD is set FALSE if GLBFIL or VPT changes.

4.14 MEFILE(UNIT, FILSEQ, NAME, TYPE, LRECL)

This function handles opening and closing files. All the arguments are integers. UNIT is the Fortran unit number. FILSEQ is the file sequence number in IBM. It is ignored here. NAME is the name in 8A4. TYPE is the operation; see the comments for the codes. LRECL is the logical record length for direct access. The value returned is 0 if there was normal; else an error code.

This routine is DEC PDP-10 Fortran-10 specific. There is a separate IBM-370 version.

4.15 MEFIL2 (UNITNO, FILNAM, KODE, ROUTNO, ERRNO)

If KODE = ZIORFN, randomize FILNAM and return it: Change chars 2-4 of it to 3 random letters. The rest of FILNAM is not changed. No tests are made for whether such a file name already exists.

Otherwise call MEFILE and then if it had an error call ERRS. If the line following the MEFIL2 call is a comment line, it will be extracted by ERRINC.TEC and put in file PRERR.DAT for ERRS to list if there is an error.
4.16 NEWSOX (ISOCK, KODE, LEAD, EOF)

This SOCKS routine initializes a socks channel to read.

4.17 PBDS(XMIN, XMAX, YMIN, YMAX)

This plotter routine scales the plot so future plot calls will have the given window transformed to fill the plot. However X and Y are scaled the same so distortion doesn't occur.

4.18 PCLEAR

This clears the plot screen.

4.19 PCLOSE

Close the plotter. On scopes, return to alpha mode.

4.20 PDRAW(X, Y)

Plot a line from the current pen position to (X, Y). No clipping is done.

4.21 PMOVE(X, Y)

Move the plotter pen to (X, Y) without plotting.

4.22 PNUMB(X, Y, N)

Draw the integer N on the screen at (X, Y). This routine, not currently used by PRISM is PDP specific.
4.23 POPEN

Initialize the plotter. On scopes, change to graphic mode.

4.24 PSTRIN(X, Y, String, Nchar)

Plot the NCHAR character string STRING in A4 format starting at X, Y (lower left corner). A table of pen motions is used so the actual plotter calls are all to PMOVE and PDRAW.

4.25 PLOTR(X, Y, N)

If N=3 call PMOVE, else call PDRAW.

4.26 PRPLOT

Read the heights for NPOL polygons from file HTFILE. Store the heights in an array at the locations determined by the hash locations of the polygon ids. The array has a zero element containing 0 to quickly handle the case of no polygon on one side of an edge. If LRANHT = TRUE, instead of reading the heights from a file, each polygon gets its hash address as a height.

Then read the NEDGE edges from the sorted file SRTFIL one by one and plot the prism map as they are read. Assume the edge coordinates are in

\[(1, \text{MXED}) \times (1, \text{MYED})\]

Assume the plot surface is

\[(1, \text{MXPL}) \times (1, \text{MYPL}) \quad \text{(integers)}\]

The minimum X plotter coordinate must be positive since it is used as a do loop parameter. Assume the heights are read as reals. Scale them so that the greatest height is PLZPCT percent of the Y dimension. If the plot isn’t the full size of the screen in either the X or Y direction, it is centered in that direction. TITLHT plot units are reserved at the bottom of the plot for titles if LDRTIT or LSTPAR = T.
The following description is obsolescent but gives an idea of what is happening.

When an edge is read, its coordinates are normalized and it is flipped if necessary to run left to right. This edge can cause up to 4 lines to be drawn on the plot; i.e. the 2 top, left and right sides of one face of a prism. First raise the edge in the Y direction by the greater of the heights of its 2 polygons to make it a top edge and repeat. Then maybe draw the left and right edges.

There are 1 or 2 top edges: at the heights of the left & right polygons above the bottom edge. The right polygon top edge defines below it a part of the top of the right prism. If the left prism is higher, there is a second top edge that defines below it (and above the right top edge) part of a side of the left prism.

The plotting is done using an horizon stored in an array in RSP that is MHorZN long. This array is initialized to 0. MHorZN must equal MXPL. The horizon starts at the bottom of the page and moves up as edges are added. The horizon divides the screen into the region below which is finished and the region above which has yet to be calculated. Although the horizon is a list of heights at MHorZN points, it is really composed of many straight line segments. You want to plot segment by segment, not point by point. Thus the heights are negated at points where the horizon changes direction. An edge becomes visible at the first point where it is higher than the horizon and invisible at the first point where it is lower. These are the elements of the horizon array that are negated.

If a top edge is added to the horizon, the value of the horizon points are raised to meet it below it. An edge is visible at a horizon point if it is at least as high as the horizon there. The reason for making edges with the same height visible is so that both edges incident at a vertex are visible. The first and last horizon locations of each visible edge segment are made negative. For a top edge, each contiguous segment of horizon all below it defines a region of the plot. Before the horizon is raised, the outline of the region (determined by the negated heights) is copied to spare space at the end of ISP. This together with the part of the top line defines a polygon that SHADE is called to shade. If the end of the edge is visible, the
polygon to be shaded has an extra side. These horizontal (in the original 3-D) edges are only drawn if LDRHOR = TRUE, but of course all the horizon calculations must be done in either case.

The left and right edges are drawn after the top and bottom edges but use the horizon values current after the bottom edge only has been drawn. They are drawn only if the edge's right polygon is lower than its left one. Since they are vertical they each use only one horizon value. They are drawn from the old horizon value to the top edge at that point. They don't change the horizon but because the top edge does one vertical edge won't be drawn twice, once as a left edge and once as a right. These vertical edges are only drawn if LDRVER = TRUE.

There are 2 types of vertical edges: those with the same prism on both sides in the plot & those with a different prism or the background on the other side. Call the latter edges SILHOUETTE edges. They are useful for outlining the prisms even if you don't want the vertical edges in general to be drawn. They are drawn if LDRSIL=.TRUE. A perimeter edge, like all vertical edges, is a left or right edge of 2 line segments. But here one segment is on the back of the prism & is invisible. The perimeter edge is the left edge of both segments or the right edge of both, while a normal edge is a left edge of one & a right edge of one.

To detect perimeter edges, use a second horizon array based at KHOR2. It is usually zero. When an edge is added to the regular horizon, all points in the second array passed over are zeroed. Then if it was a top edge, the loc in KHOR2 of the left endpoint gets the negative of the old first horizon's value there and the loc of the right endpoint gets the positive old horizon value at its loc in KHOR. Thus all the left & right endpoints on the horizon are identified with their old horizon values. However before I set the new values, I check for whether there is already an endpoint there & if so is it of the opposite type. If it is, I have a perimeter edge to draw.

How the shading works is described in the users' manual.
There is a problem aligning the shading where the edges meet especially if the edges are short compared to the shading spacing. To fix this, the spacing is changed slightly so the edge contains an integral number of shading lines. The number of shading lines is changed to either the next lower or the next higher integer randomly so that the expected darkness is unchanged. Thus a short edge may end up with no shading lines.

The vertical shading lines are drawn as the horizon is raised for the edge. They go from the old to the new horizon value. Alternate lines are drawn in opposite directions to save plotting time.

If LCONTR = TRUE, horizontal contour lines are drawn on the sides every COSPAC units, starting from 0. This is done by drawing several horizontal edges, raising the horizon for each one, instead of just the bottom & top edges.

If LABEL = TRUE, the polygon titles as read from the values file are plotted at the centroid of each polygon. The centroids were written to the sorted file by READCH. No action is taken to see that the title is inside the polygon or that that part of the polygon top is visible. To save space, if LABEL = FALSE, the title and centroid arrays are not allocated. This allows maps with more polygons to be plotted without increasing ISP's size. If LRANHT = TRUE, blank labels are used.

Finally draw a box around the plot (if LDRBOX = TRUE) and the titles (if LDRTIT = TRUE). There 4 lines of titles:
1- a user supplied title,
2- the date & time & user name,
3- the chain file title,
4- the heights file title (or a message if no heights file).
If LSTPAR = TRUE, a fifth line lists various parameters of the plot. This is useful say for comparing the effects of different shading. Then go into alpha mode to save the screen. Wait for a CR before continuing.
4.27 RANNUM (DUMMY).

This function returns a real number from 0 to 1. It is quick and dirty and repeats when the program is restarted but this is OK since it is only used for shading. It is:

\[ I = \text{MOD}(I*44201, 45307) \]

4.28 RDVEC(IUNIT, V, N, EOF)

Read from unit IUNIT an unformatted vector V that is N words long. Set Logical EOF to TRUE if an eof is read.

4.29 READCH (EDGFIL, NEDGE)

Assuming that CHNFILE is an Odyssey GBF file, either binary or formatted as specified in GLBFIL, and GLBFIL is its global file, read it and write it reformatted to EDGFIL. Return the number of new edges in NEDGE. The edges in the chains are written one record per edge. Create file SRTFIL that will eventually receive the sorted edges & write header info into it such as NPOL, NPOLAR, the polygon id hash table and title of the chain file. Leave this file open on unit IOSORT which was set by the caller. Return NPOL as the number of polygons and NPOLAR as the array's length. Since the array is hashed, NPOLAR = 1.5*NPOL.

The viewpoint is defined in modified radial coordinates. The projection is 2 point perspective from the viewpoint to the centre of the box around the plot. The bounds read from the global file are enlarged by 0.001 of their extent in either dimension to compensate for their possible slight inaccuracy.

\[
\begin{align*}
X' &= X \cos(\text{VPANGH}) + Y \sin(\text{VPANGH}) \\
Y' &= \left( -X \sin(\text{VPANGH}) + Y \cos(\text{VPANGH}) \right) \cos(\text{VPANGV}) \\
X'' &= X' / (1 + Y' / \text{VPDIST}) \\
Y'' &= Y' / (1 + Y' / \text{VPDIST})
\end{align*}
\]

Given the projection, scaling factors XR, XD, YR & YD are chosen so that the coordinates will be in (1, HALFHI). They will touch the bounds if the chain file touches the 4
corners of its box. Since in general it doesn't, MAXXED, MAXYED, MINXED & MINYED are set to the bounds of the scaled plot.

\[ 1 \leq \text{MINXED} < \text{MAXXED} \leq \text{HALFHI} \]

and the same for Y. At the end, debugging stats such as the average & max chain length are printed if desired.

As the chains are processed, the centroids of the polygons are calculated in arrays bases at KCX & KCY. They are later used for labelling the prisms.

Edges of length 0 after transformation and conversion to integers are not written and not included in NEDGE.

4.30 READG(IOGLOB, IOTALK, PRINT)

Read a globals file from unit IOGLOB. If logical PRINT is TRUE, print the values, nicely labelled and formatted on unit IOTALK. IOGLOB is assumed to be open on entry and is left open. The values are returned in common block GLOBAL which is described in the comments for READG.

4.31 RECALO (KFREE, LENFRE, RECLEN, K, KODE)

Manage a free space based in ISP at KFREE that is LENFRE words long by allocating and freeing records that are RECLEN words long. RECLEN must not change once this is initialized. Uses a linked list.

If there is only one active area at a time, NRALO and HIRALO are the current number of records and the greatest ever number of records in the currently active area.

KODE ACTION
ZRAINI Initialize the free space. (Required at first).
ZRAALO Allocate a record & return its base in K.
ZRAFRE Free the record based at K.
If INTERN(13) >= 2, blocks are never actually freed. Since they are therefore never reused, debugging is easier.

4.32 SBLOCK (KODE, ICHAN, FILNAM/REC, UNIT/K, RECLEN/0)

Block a sequential file with short records. Do not span records over block boundaries. You are responsible for setting BLKSIZ in common. You can do I/O independently on 2 channels. Each block is padded to BLKSIZ on writing. This padding is ignored on reading. SBLOCK cannot be overlaid since it has internal info that must be preserved from call to call. SBLOCK doesn't alloc or free unit numbers (as with IOALO).

For speed, I handle 1 word records separately if KODE=ZBLREA or ZBLWRI.

To detect an EOF in the middle of the last block, when writing I pad it with INTBIG in the remaining words. If the first word of a genuine record were INTBIG, you will get a premature EOF there.

How to call SBLOCK:

(ZBLIRE, ICHAN, FILNAM, UNIT, RECLEN). This opens and inits a file for reading.

(ZBLIRO, ICHAN, FILNAM, UNIT, RECLEN) This inits for reading a file that is already open and presumably has had some header info read.

(ZBLIWR, ICHAN, FILNAM, UNIT, RECLEN) This creates and inits for writing a new file.

(ZBLIWO, ICHAN, FILNAM, UNIT, RECLEN) This inits for writing a file that is already open and probably has had some header info written.

(ZBLREA, ICHAN, REC, K, 0) Read a record. If EOF is reached and K.NE.0, set K to -1. Else stop.
(ZBLWRI, ICHAN, REC, 0, 0) Write a record.

(ZBLCLO, ICHAN, 0, 0, 0) If writing, write the last buffer and in any case close the channel.

**ARGUMENTS:**

ICHAN The channel number (1 or 2).
FILNAM Int. The file name, 3A4.
UNIT Int. I/O unit #.
RECLEN Int. Record length, words.

4.33 SHADE(BASE, NPTS, POLID, KODE)

Shade a region of the plot.

BASE (Int) The base in ISP of the 2-D array of vertices of
the region. They are integers in plotter coordinates.
NPTS The # of vertices in the region.
POLID The i.d. # of the original polygon whose prism this
polygon is a part of.
KODE 1 This region is on the top of the prism.
2 ... on the side.

SHADE is called separately for every disjoint region comprising each face of the prism.

4.34 SCNCMD(STRING, STRLEN, DEVICE, FILE, EXT, PPN)

This PDP-10 specific routine takes an A5 format STRING,
STRLEN words long, that is assumed to be a file name in the
format DEV:FILE.EXT[PP,N] and parses it, returning the
pieces. DEVICE and FILE are returned 2A5. EXT is A5 and
PPN is 2I.
4.35 SLAY(I)

This is called by the SOCKS routines in case of a terminal error. It prints I and stops.

4.36 SORT(KB, RECLEN, NREC, KK)

Shellsort an integer array of NREC records, each of length RECLEN (<=10). The array is based at KB. Even when RECLEN =1, KB bases a 2-D array. I.e. the array is ISP(KB+1*RECLEN+J), J=1,RECLEN, I=1,NREC). If KK >0, KK bases an array that is returned with the new location of each old element. For speed, handle the case RECLEN=1 separately. This goes a lot faster since comparing and moving records is just 1 operation instead of a DO loop.

4.37 SRTEDG (INFILE, NEDGE)

Assume an edge file of NREC records named INFILE. It is initially sorted by minimum Y coordinate of each edge. Write a file named SRTFIL with the edges sorted so that a high numbered edge never overlaps in front a lower numbered edge when looking from the low Y direction. SRTFIL is received already open on unit #IOSORT with some header info already written.

Method: Define the edge's endpoints as (X1,Y1) & (X2,Y2) where Y1 <= Y2. Set up a plist (heap) and a binary search tree. A plist (priority list) is a data structure that has 2 operations: add an element and return & delete the smallest element. Here I implement the plist by a heap. Read INFILE, edge by edge. Edges are read, held for a while and written. At any time, the active edges are those intersected by a scan line: Y=YSCAN. Each active edge is in both heap and tree. The heap contains pointers to the active edges, sorted by Y2 so that the edge with the lowest Y2 can be returned fast. The tree contains pointers to the active edges sorted by where the edge intersects the scan line. This value changes but since the edges never cross, no 2 edges in the tree ever change order.
Each edge, E, has 2 other attributes: 1) An ordered list of edges that E hides, & 2) A count of the number of hidden lists that contain E. The edges in the hidden lists are still in core - they have been read from INFILE but not yet written to SRTFIL. However they are not in the heap or tree. Call them semiactive edges. They are only accessible through the 1 or more hidden lists that each is on. Hidden lists form a tree structure since a semiactive edge can have a hidden list of its own. There is never any circularity (Exercise: Prove it.) Edges on E’s list all have Y2 less than E’s and they are ordered by increasing Y2. Thus if E is drawn, followed by its list in order, there will be no conflicts among them. The scan line is above the semiactive edges.

When an edge is read, add pointers to it to the heap and tree. However, if its Y1 is higher than the lowest (the first) Y2 in the heap, put the new edge aside while all the lower points in the heap are processed. Take the first edge in the heap. Call it the low edge. The low edge, call it E, will either be processed & written or changed to semiactive and put on another edge’s list. The reason is that E cannot be written if it is hidden by any remaining active edge. It is sufficient to test against the 2 adjacent edges in the tree (the one to the left & the one to the right). The reason is that if the edge immediately to the left of E doesn’t hide it then no edge to the left does, & the same for the right. If neither hides E, process it. Else add E to the end of the hidden list of whichever (or both) of the adjacent edges hides it. Also set the count of the number of lists it is on.

Edge KE hides edge KLO when: The X coordinates must have at least 1 point in common. Then take the average of the middle 2 of the X coords of the 4 endpoints. Find the Y values, YE & YLO, of the edges at that point. If either edge is vertical, take the Y of its higher endpoint. Then YE < YLO. This handles the case of KE & KLO having a common low endpoint while one of them is vertical.

When an edge is processed & written, its hidden list must be handled. The following definitions are recursive:

WRITE an edge: Send it to SRTFIL, decrement the hidden count of all edges on its list, process its hidden list and delete it from the edge space.
PROCESS a hidden list: Go down the list in order. Write any edge that has a hidden count of zero.

Thus this is a depth first walk of the original edge's hidden list.

When the above has been repeated until there are no more qualifying 10 edges on the heap, add the waiting new edge to the heap & tree. At the end of INFILE, an imaginary edge with $Y_1 = \infty$ causes the remaining active edges to be written.

Format of the edges (active or semiactive) in edge space: (8 word records)

1 $X_1$
2 $Y_1$
3 $X_2$
4 $Y_2$
5 LPOL/RPOL (hash addresses)
6 Ptr to hidden list, or 0.
7 Ptr to last block of hidden list, or 0.
8 Hidden count.

The hidden list is a linked list of 8 word blocks. Thus they can be allocated and freed like edge records.

1-7 Edge bases, or 0.
8 Ptr to next block, or 0.

4.38 TEXT( X, Y, AHITE, AWIDTH, ANGLE, STRING, NWRD, KTYPE)

TEXT plots a string at the given location, size and angle using only plotter line motions by calling PLOTTR. It is better documented by its comments.

4.39 TIMES ( N )

Keep track of how many times subroutine #N is called, and how much time it uses. TIMES must be called on entry (CALL TIMES (N) ) and on exit ( CALL TIMES (-1) ). The routines may be nested up to 99 deep. Also, optionally
trace calls to and returns from timed routines.

Bump RNCALL everytime TIMES is called. Then then when
IRUNTI returns the CPU time, lower that time by RNCALL *
RTCORR to hopefully negate the time used in calling TIMES
itself.

If INTERN(4) .NE. 0, list all calls to TIMES. On a
call on entry to a routine, list the stack of open routines.
TIMLST(1)=4 and TIMDEP=2 initially so that TIMES is always
at the bottom of the stack to make things easy.

4.40 TIMINI

Call routine TIMES 2*100 times to find out how long it
takes and set RTCORR accordingly.

4.41 TIMPRT

Print the timing & frequency stats for those routines
called at least once (and that called times) sorted by
routine name.

4.42 TRECOM (KA, KB)

TRECOM is used to order the TREE. Compare the edges
based at KA & KB and return -1, 0 or 1 according as the
first is smaller, equal or larger. If KA=KB assume the
edges are equal. Otherwise the comparison is based on the X
coordinates of the 2 edges at Y=YSCAN which is X1 for the
new edge. There is a messy case when 2 edges being compared
have the same X values. Since edges never cross (assumption
on the initial data) but frequently have common endpoints
and then diverge, their order is important. It is their
order above YSCAN that is wanted and this is gotten by using
their slopes. There are different cases depending on the
slopes' signs. The setting the horizontal slope infinite
when the horizontal edge is vertical is only used later if
it has a common X with the other edge. It might save time
to store the edge equations instead of calculating them each
time but compared to the setup time of calling TRECOM etc. it is probably insignificant.

There is a special case when the edges are coincident. Since \( \text{SIGN}(1,0) \) is \(-1\) or \(1\) randomly, I arbitrarily set the edge at the lowest address to be the left one.

4.43 TREE (KTREE, TRELEN/KREC, \(0/\text{LOC}\), KODE)

Implement a search tree holding pointers to records. It is stored in \((\text{ISP} (\text{KTREE}+I), I=1,\text{TRELEN})\). \(\text{KREC}\) is the base of the record given or returned. The record itself is in \((\text{ISP} (\text{KREC}+I), I=1,\ldots)\). After an add or delete, \(\text{LOC}\) is the record's number (or \(0\) if it doesn't exist). Many trees may be active at once. The first \(\text{LTRDAT}=2\) locations of the tree are used to store \(\text{TRELEN}\) etc. so only \(\text{TRELEN} - \text{LTRDAT}\) elements are allowed in the tree. The \(\text{LOC}\)-th element of the tree is

\[ \text{KREC} = \text{ISP}(\text{KTREE} + \text{LTRDAT} + I). \]

The initial call is:

\[ \text{CALL TREE(KTREE, TRELEN, 0, ZTRINI)} \]

Succeeding calls are:

\[ \text{CALL TREE(KTREE, KREC, LOC, KODE)} \]

**KODE**

**OPERATION**

\[ \text{ZTRINI} \quad \text{Initialize tree. required at first.} \]

\[ \text{ZTRADD} \quad \text{Add REC.} \]

\[ \text{ZTRDEL} \quad \text{Delete REC. Return LOC=0 if not found.} \]

\[ \text{ZTRLOC} \quad \text{Locate REC and return its relative number in the tree in LOC.} \]

\[ \text{ZTRLEF} \quad \text{Return in LOC the base of the largest element smaller than REC in the tree, or 0 if REC is the smallest.} \]

\[ \text{ZTRRGT} \quad \text{Return the smallest element larger than REC.} \]

For better error detection, adding an existing element, or finding left or right neighbours or deleting a nonexisting element are terminal errors. You can remove these traps for flexibility if you want.
4.44 WTVEC(LU, VEC, N)

Write the unformatted vector VEC(N) to unit LU.

5.0 CALLING TREE

MAIN
TIMINI
COSTS

USAGE
GMTDT

NEWPAG
DAYTIM

DATE
TIME
A5TOA1
A1TOA4

LINGO2
LINGO

READCH
JOIN
HASH

ITTOME
SLOY
RDVEC
XFER

NEWSOX
SLOY
RDVEC
XFER

READG
SBLOCK
RDVEC

EXSORT
SORT

INVPER
RDVEC
WTVEC
MEFIL2
PRISM PROGRAM LOGIC MANUAL
CALLING TREE

SRTEDG TREE TRECOM
HEAP0 RECALO
SBLOCK RDVEC

PRPLOT HASH
SPLIT RANNUM
RDVEC
PCLEAR NEWPAG
PCLOSE ANMODE
PDRAW DRWABS
PMOVE MOVABS
PNUMB PMOVE
ANMODE
GRMODE
PSTRIN TEXT
GVEC
PLOTR PMOVE
RDVEC PDRAW
SBLOCK SHADE

MEFIL2 SETRAN
TIMPRT

PLOT stands for all the plotting routines. Everyone
calls ERRS and TIME.

Don’t overlay RANNUM deeper than PRPLOT.
6.8 MACHINE DEPENDENCIES

1. BLKSIZ, the size of buffers though bad values should work inefficiently.

2. The file handlers MFILE & MFILE2.

3. The plotting routines, although the interface should remain the same.

4. During debugging, PRISM calls the Tek routines GRMODE & ANMODE. They can later be dummied or ignored.

5. The real and integer arrays ISP & RSP are equivalenced and values in any word may be set to both real and integer interchangeably. Thus ISP & RSP cannot be optimized. I don’t know of any machines this would cause a problem on.

6. Subscripts more complex than (C1*V +C2) are used but the expressions never contain parentheses.

7. Array parameters in subroutines may be dimensioned (1) for efficiency.

8. The units and filenames mentioned in the UNITS & FILENAMES section are probably machine dependent.

9. Also the highest legal values for numbers in BIGNUM, HALFHI & INTBICG although the current values are OK for both PDP-10 & IBM-360. These numbers can be smaller than the actual maxima as long as they are pretty big.

10. Various machine dependent routines, DATE, TIME, UPNAME, IRUNTI, SETDEV are used only for titles and diagnostics. They may be dummied.

11. ‘END=’ is used in most READ statements.
7.0 COMMON BLOCKS

COMCOS contains the local variables of routine COSTS so it can be overlaid.

EXTINT contains character codes used for converting characters from ASCII to an internal format.

PCOM One big block with all the global variables in the parts of PRISMMAP that I wrote.

PLOCOM contains the plotter scaling and location numbers.

SOCK Global variables used by SOCKS.

STORGE Contains a big array, ISP that storage is allocated from, dynamically. Also used by SOCKS.

OPATH, LEXS, IFSTACK, STACK, PILES, VAULT, GUARD, PURITY, INTEXT and UNITS are used by the command language parser. Until it is implemented, they can be deleted.

8.0 COMMON VARIABLES IN PCOM

BIGNUM The biggest real number allowed (or somewhat less).

BLKSIZ Length of buffer for optimal unformatted IO. If it is made bigger than 127, arrays in SBLOCK must be increased.

BUSUNI (maxuni) LOGICAL. Is this unit in use?

CHNFL Input chain file.

CMT (20) User supplied plot title.
CONHTS  Spacing of contour lines, user units.
CONPLO  Spacing of contour lines on sides of prisms, plotter units.
GLBFIL  Input globals file.
HALFHI  The biggest integer allowing 2 to be packed in 1 word.
HIHEAP  The greatest size of any heap that has been added to since the last heap init.
HIISP   Max # elements in ISP used. This ignores that EXSORT and PRPLOT use all available space, ISP should be allocated at least 1K bigger than this.
HIRALO  Max # records alloc at 1 time by RECALO. This is correct when there is only 1 active RECALO area.
HISPAC  The farthest spacing of shading lines, in plotter units.
HITREE  The greatest depth of any tree element accessed in a tree since the last tree init.
HTFILE  Input values file.
INTBIG  The biggest legal integer (somewhat smaller is OK.) However this must be bigger than the largest combined word (formed by joining 2 halfword coordinates etc.) This is also used as a filler in EXSORT & SBLOCK so its appearance in genuine records there will cause an error.
INTERN  (mrount) controls printing of internal info in each routine. 0 = print none.
IOCHN   I/O unit.
IOCMD   I/O unit.
IOEDGE  I/O unit.
IOERR   I/O unit.
IOGLB   I/O unit.
IOHT    I/O unit.
IOMSG   I/O unit.
IOSORT  I/O unit.
IOTSOR  (6) I/O units.
IOTTY   I/O unit.
IPLOTR  Type of plotter:
        1- Tek 4014
        2- Tek 4010
        3- Calcomp
ISP     A big array that almost all other arrays are allocated from. This gives effective dynamic storage allocation. RSP is equivalenced to it. So are various small arrays.
KSHADE  The vertical shading type.
2 Vertical lines spaced according to LITANG.
LCONTR  Draw horizontal contour lines on sides of prisms?
LDRBOX  Draw a box around the plot?
LDRHOR  Should horizontal (in the 3-D prism) lines on the
plot be drawn?
LDRSIL  When drawing vertical edges, draw only silhouette
edges?
LDRTIT  Draw titles at the bottom of the plot?
LDRVRE  Should vertical lines (the sides of the prisms) be
drawn, if visible?
LHEDAT  Length of fixed data area at start of heaps.
LITANG  (Real) The angle of illumination for shading,
degrees relative to VPANGH, positive rotation.
LLOHTZ  Should the heights be displaced so the lowest is 0?
LABEL   Should the tops of the prisms on the plot be
labelled?
LFORMA  Is the chain file formatted (T) or binary (F)?
LOS PAC  The closest spacing of shading lines, in plotter
units.
LRANHT  Use random heights instead of reading a heights
file?
LSTPAR  List parameters at bottom of plot?
LTIME   TRUE => keep frequency and timing stats for
routines.
LTRDAT  LENGTH of fixed data area at start of trees.
MACHIN  Type of computer this is being run on:
1 - PDP-10
2 - IBM-360/370
3 - Univac 1108
MAXTMP  Size of TMPNAM.
MAXUNI  Highest legal unit number.
MAXXED  Highest X edge coordinate (internally, after
normalization).  <= HALFHI.
MAXYED  Highest Y edge coordinate.  <= HALFHI.
MCMT    Size of CMT, chars.
MINXED  Lowest X edge coordinate.  >= 1.
MINYED  Lowest Y edge coordinate.  >= 1.
MISP    Size of array ISP.
MROUTS  # of routine names that can fit in array RTNAME.
MSGFIL  (3) Name of file containing error messages.
MXPL    Highest X plotter screen coordinate.
MYPL    Highest Y plotter screen coordinate.
NCMT    # chars in CMT.
NHEAP  Current size of heap.
NISP   # of words of ISP in use now.
NRALO Current # of records allocated by RECALO; correct when there is only 1 active RECALO area.
NRoutes # routes.
PLZPCT On the plot, the percentage of the Y extent that the greatest height is to be.
PXMAX |
PXMIN \ Bounds of plotter window.
PYMIN /
PYMAX / RAFFREE A pointer to the list of free records in RECALO.
RHASH The factor that the # of polygons is multiplied by to give the size of the polygon hash table. 1.5 is conservative.
RNCALL # calls to TIMES since timing was initialized.
RTCORR CPU time used by 2 calls to TIMES.
RTIME Time used in each routine, exc.
RTIME2 Time used in each routine, inc.
RTNAME Names of routines.
RTNUMB # times routines have been called.
RTNUM2 # times a routine has called other timed routines.
SBLEN (2) Length of a record for each SBLOCK unit.
SBNAME (3,2) Names of the 2 SBLOCK files in 3A4 format.
SBUNI (2) # of each SBLOCK unit.
SIREC (2) When an SBLOCK unit is being read, the number of records read so far.
SNREC (2) # of records in an SBLOCK file.
SRTFILE Input or output sorted edge file.
TIMDEP Height of stack of open routines in TIMES.
TIMLST (50) The stack of open routines in TIMES.
TITLHT Space to be reserved at bottom of plot for titles, plotter units.
TMPNAM Names of temporary files in 3A4 format. The first 6 are used for sorting. #7 is the edge file made from the input chain file. #8 is the sorted edge file.
TRNLOC # times elements have been located by TREE since last init.
TRSLLOC Sum of # elements in TREE each time it was called since last init.
VERSIO Version # of PRISM. The hundreds digit is the version # of the sorted edge file format.
VPANGH Horizontal angle of rotation of viewpoint from (0, -inf) in degrees to the right.
VPANGV  Vertical angle of rotation of viewpoint in degrees relative to horizontal.
VPDIST  Distance of viewpoint from origin, in data coords.
VPT     (3) equivalenced to (VPANGH, VPANGV, VPDIST).
YSCAN   Current scan line value in SR TEDG.
ZXXXYY  Codes for routine abbreviated XX. Except for the ZIO codes which are passed directly to MEFIL2, the codes are big random numbers to trap errors better.
ZBL??  Codes for SBLOCK.
ZHA??  Codes for HASH.
ZHE??  Codes for HEAP0.
ZIO??  Codes for MEFILE.
ZLI??  Codes for LINGO.
ZRA??  Codes for RECALO.
ZTR??  Codes for TREE

9.0 DATA FILE FORMATS

Each I/O op logically moves 1 block (usually 127 words). For efficiency, the block size should be near the actual hardware block size. Each block is usually divided into constant length records. If it weren't for efficiency, the record would be the unit of I/O. In the following, fields between / are in 1 word. The files in chronological order are:
9.1 Global Chain File

See the comments in routine READG.

9.2 Chain File

The chain file can be formatted or binary. Formatted is machine dependent while binary takes only half the space and is more efficient to read and write. Whether the chain file is formatted or binary and whether the points include deviations is known from flags in the global file.

9.2.1 Formatted

Each chain is 2 or records. The first is (6I10), the fixed part. The rest contain the points in (6E12.5) format. For every point there is X, Y and possibly a deviation.

9.2.2 Binary

This Fortran unformatted file is in blocks of LONGS=510 words. It is divided into spanned logical records. There is a pair of records for each chain:

Fixed record - NFIX(I)=6 integers:

1. Chain id.
2. # points in the chain & pieces in the variables record.
3. From node id.
4. To node id.
5. Left polygon id.
6. Right polygon id.
Variable record - JAMVER(ISOCK)=2 or 3 words/point. The floating point coordinates of each point & maybe the detail level.

9.3 Heights File(s)

See the description in the users’ manual.

9.4 Edge File # 1 (partly Sorted)

Each record describes 1 edge of the chain file: (X1,Y1), (X2,Y2). Assume that Y1 <= Y2. Else swap the edge endpoints and adjacent polygons to make it true. Then the record is:

\[
Y1,X1 / Y2,X2 / LPOL,RPOL
\]

The coordinates are integers and range from 1 to HALFHI. This file is written unsorted then sorted by Y1. LPOL and RPOL are the hash addresses of the polygons on the left and right sides of the edge (0 if no polygon). Hash addresses are used since polygon numbers are sometimes bigger than a halfword. The polygon numbers are retrievable from the hash table.

9.5 Edge File # 2 (sorted)

Unformatted Fortran records:

1st record, 8 ints & an array:

1 VERSIO
2 NEDGE
3 NPOL
4 NPOLAR
5 MAXXED
6 MINXED
7 MAXYED
8 MINYED

9 Title of chain file, 20A4..
PRISM PROGRAM LOGIC MANUAL
DATA FILE FORMATS

B-79

2nd record, 3 arrays of NPOLAR ints each:

1 The polygon id hash table.
2 The X coords of the polygon centroids.
3 The Y coords of the polygon centroids.

Succeeding records. Each record stands for 1 edge:

X1,Y1 / X2,Y2 / LPOL,RPOL

1 <= X1,X2 <= MXED
1 <= Y1,Y2 <= MYED
LPOL = Hash addr of left polygon.
RPOL = Hash addr of right polygon.

These edges are written sorted by visibility.

9.6 Error File

This formatted file contains the meanings of the error
codes passed to ERRS and MEFIL2. It is produced by TECO
macro ERRINC.TEC from comment lines after the calls to those
routines. The format is (I3, 1XI3, 1X18A4) for the routine
number, error number and message. It is searched linearly
until the routine and error numbers match.
**PRISM PROGRAM LOGIC MANUAL**

**UNITS & FILES**

**10.0 UNITS & FILES**

Columns:
1: Unit name.
2: Default unit number.
3: In or Out.
4: Variable(s) containing file name(s).
5: Description

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOTTY</td>
<td>5</td>
<td>I/O</td>
<td></td>
<td>TTY:</td>
<td>To read commands from.</td>
</tr>
<tr>
<td>IOCMD</td>
<td>5</td>
<td>I</td>
<td></td>
<td>MSGFIL</td>
<td>To write messages to.</td>
</tr>
<tr>
<td>IOMSG</td>
<td>6</td>
<td>O</td>
<td></td>
<td>File of error messages.</td>
<td></td>
</tr>
<tr>
<td>IOCHN</td>
<td>2</td>
<td>I</td>
<td></td>
<td>GLBFIL</td>
<td>User supplied input files.</td>
</tr>
<tr>
<td>IOGLB</td>
<td>13</td>
<td></td>
<td></td>
<td>CHNFIL</td>
<td></td>
</tr>
<tr>
<td>IOHT</td>
<td>14</td>
<td>HTFILE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOSORT</td>
<td>3</td>
<td>I/O</td>
<td></td>
<td>SRTFIL</td>
<td>The sorted edge file, if not user supplied then returned by PRISM.</td>
</tr>
<tr>
<td>IOTSOR</td>
<td>7-12</td>
<td>I/O</td>
<td>TMPNAM(<em>,1)-(</em>,6)</td>
<td>Temporary sort files for EXSORT.</td>
<td></td>
</tr>
<tr>
<td>IOEDGE</td>
<td>4</td>
<td>I/O</td>
<td></td>
<td>TMPNAM(*,7)</td>
<td>Input &amp; output file for EXSORT.</td>
</tr>
</tbody>
</table>

All file names are 8A4. TMPNAM is set randomly in MAIN. So is SRTFIL if you don’t specify it. IOTTY is only to wait for a prompt before & after plotting.
11.0 LIBRARY Routines Used

1. Standard FORTRAN library routines are called.

2. LCG10.MAC contains various short machine dependent routines. [3111,742] 14-Jan-77.

3. MEFILE.F4 - File opening, closing and deleting. [3111,742] 14-Jan-77.

4. SOCKS.F4 - Chain handling routines such as RDVEC, WTVEC, NEWSOX and OLDSOX.

5. GLOBAL.ODY.

6. CRUSTY.CMD[3111,1250] parser routines. (compiled to LING10.REL[3111,13111]).

7. PLOT10.MAC - [3111,13111].

8. LCG10B.MAC - various short pretty-printing routines such as IRUNTI, GMTDT, UPNAME, USAGE.

9. DEC PDP-10 routines: DATE, TIME, RAN, SETRAN. SETRAN & RAN are only used to produce a random TMP file name in MEFIL2 on the PDP-10.

12.0 USING SOCKS ROUTINES

This is not a complete description but only covers the parts relevent to PRISM.
12.1 General Notes

1. There are 2 sequential files for each database, one with some global info & the other with the data: e.g. CLEV.G.DAT & CLEV.E.DAT.

2. They use commons SOCK & STORGE.

3. Up to 3 chain files can be open at once.

4. ELF can be used to extract parts of chain files.

12.2 How To Use

1. Open the global file and read it with READG if you want various important variables.

2. Initialize common variables such as IOSOX(I) of the unit number of the I-th channel. Set NFIX(IOSOX) to LENCHD from the global file.

3. Call MEFILE to open the chain file on unit IOSOX(I).

4. Call NEWSOX(ISOCK, KODE, LEAD, EOF) to set up buffers & return header record.

ISOCK(I)       Channel no.
KODE(I)        0-read, 1-write.
LEAD(I/O)       Obsolete argument now dummy.
EOF(O)          Set if eof reading header.

5. Call ITTOME(ISOCK, KNEE, LEG, EOF) to return 1 chain.

ISOCK(I)       Channel number.
KNEE(I) (O)    Fixed record.
LEG(I) (O)     Variable record.
EOF (O)        TRUE => at eof.

6. Call OLDSOX(ISOCK) at end to close things.
12.3 Existing Chain Files

are listed in [3111,13111]FILES.DOC.

13.0 SIMPLE STATISTICAL ANALYSIS

Assume that

\[ N = \text{# polygons (say 1000)} \]
\[ K = \text{# sides each (say 100)} \]

If they polygons form a square grid, \( \sqrt{N} \) on a side, then

\[ \text{AEL} = \text{average edge length} \]
\[ = \frac{1}{K} / \sqrt{N} \]

\[ \text{NE} = \text{# edges} \]
\[ = 2 \cdot N \cdot K \]

\[ \text{TEL} = \text{total edge length} \]
\[ = \text{NE} \cdot \text{AEL} \]
\[ = 2 \cdot \sqrt{N} \]

(in fact it would probably be greater since the sides of the polygons are curved, not straight). In 1-D, there is a 2*L chance of 2 edge, each of length L, intersects when randomly distributed in a line 1 long (L << 1). Thus

\[ \text{PX} = \text{probability of an intersection} \]
\[ = 2 \cdot \text{AEL} \]
\[ = 2 / K / \sqrt{N} \]
\[ = 1500, \text{ say} \]

\[ \text{NX} = \text{# edges intersecting a given edge} \]
PRISM PROGRAM LOGIC MANUAL
SIMPLE STATISTICAL ANALYSIS

\[(\text{in 1-D})
\]
\[= \text{NE} \times \text{PX}
\]
\[= 2 \sqrt{\text{N}}
\]
\[= 6\theta\]

This is the number of edges intersecting a given edge when projected onto the Y axis. It ignores complications like foreshortening due to angles of rotation. It is approximately the average size of the tree in SRTEDG.

14.0 THINGS TO DO & THOUGHTS FOR THE FUTURE

1. Add chain pieces, not edges, to heap. This might mean variable length EXSORT & sort.

2. Put in stats count of chains changing direction.

3. How to make EXSORT variable length:
   1. Pad everything to max length.
   2. Have several consecutive fixed length recs with all but the first flagged to remain adjacent to the first.
   3. Pointers & dynamic allocation.
   4. Pointers or padding the only ways for SORT.

4. Do 3 point perspective.
5. Better TREE routine, maybe in Macro.
6. Get crosshatch from VPLT.

15.0 DISK FILES

This section is only relevant for Aiken.

1. PRISM0.REL The rel file for the Macro routines used by PRISM. It currently contains LCG10, LCG10B & PLOT10.
2. LING10.REL[3111,13111] parser routines.
3. LCG10.MAC Short necessary Macro routines.
4. LCG10B.MAC Short unnecessary but nice Macro routines.
5. PLOT10.MAC Tektronix plotter package, modified.
6. PRISM.FLX The source code for all the Fortran or Flecs PRISM routines.
7. PRPLM.RNO The programming logic manual.
8. PRMANL.RNO The users' guide.
9. PRERR.DAT The error messages. Only used when PRISM has a fatal error.
10. ERRINC.TEC A Teco macro used to produce PRERR.DAT and also sequentially number error calls in PRISM. See description of routine ERRS.
11. PIBM.CTL  A control file to prepare PRISM for IBM.
12. PIBM1.TEC  A teco macro to help converting PRISM to IBM.
13. PIBM2.TEC  Another.
14. POVL.CTL  A batch control file to link P.REL and PRISM0.REL to make an overlaid version.
15. POVL.CMD  The LINK input needed for overlaying.
16. PGLIB.DAT  The language definition.
17. PLAN.DAT  The language tables written by GLIB.
18. PRICOM.FLX  The main common block.

16.0  INCLUDING PRISM IN OTHER PROGRAMS

Let your program serve the functions of PRISM's main routine. You can delete routine LINGO and LINGO2 and all common blocks they use except EXTINT which is also used in PSTRIN.
. 17.0 OVERLAY STRUCTURE

The following is the overlay structure used on the PDP-10. It should work on other systems, although depending on the exact routine lengths, refinements may be possible.

The structure is a tree with a root and 4 sons. The root is everything that is not in one of the sons. The fourth son is the biggest but its largest routines are called in a line of descent thus: PRPLOT -> TEXT -> GVEC so they cannot be overlaid (at least until we get better link editors). The sons are:

1. ITTOME, NEWSOX, READCH, READG, SLAY.
2. EXSORT, SORT.
3. HEAP0, RECALO, SRTEDG, TRECOM, TREE.
4. GVEC, PCLEAR, PCLOSE, PDRAW, PMOVE, POPEN, PSTRIN, PLOTR, PRPLOT, RANNUM, SHADE, TEXT, (all plotter routines).
APPENDIX C

GLOSSARY

BACK FACE: A face on the back side of a closed polyhedron; that is one whose inside face faces the viewpoint and whose outside side points away.

CENTERPOINT: The point on the perspective plane that is the foot of a line dropped from the viewpoint. This is the closest point on the perspective plane to the viewpoint. It appears on the center of the screen.

CENTRAL POINT: of a well tesselated polyhedron is any point within the polyhedron from which every face is completely visible.

CHAIN: An ordered sequence of points in the plane that represents an approximation to a curved line such as a coastline. If the chain is a boundary of some sort, it also has the numbers of the polygons on each side as attributes. Another attribute is its accuracy of representation.
CLIPPING: The process of intersecting a scene with a box in 3-D and discarding those parts outside. Some edges and faces will be eliminated and others cut. New vertices must be created for these edges new endpoints.

CONVEX POLYHEDRON: A line joining any two points inside a convex polyhedron is totally inside the polyhedron.

DIRECTION COSINES: In geometry, assume a ray, OR, from the origin. Its direction cosines are the cosines of the angles it makes with the three axes. They uniquely define OR's direction.

EDGE: A straight line in the scene between two vertices, that is defined by the numbers of those vertices. Edges also delimit faces. Some edges may exist just to join a hole in a face to the outer boundary and so shouldn't be drawn.

FACE: A planar polygon in 3-D in the scene that is defined by the ordered list of the numbers of its vertices. It must not intersect itself and if it has an island or hole, that must be connected to the outer perimeter by edges.

FRACTIONAL DIMENSION CURVE: A set of points in the plane that under a suitably generalized notion of "dimension" appears to have a dimension between one and two. These
curves are the limits of infinite sequences of ever more complex curves.

GEOGRAPHIC BASE FILE (GBF): This is actually a pair of files and defines a digitized map where polygons are delimited by boundaries that are chains. The GLOBALS file contains global information such as title, date, number of points, number of chains, etc. The CHAIN file is an alternating sequence of fixed and variable length records. Each (fixed, variable) pair defines one chain. The fixed record contains information such as the number of points in the chain and its left and right polygons. The variable record contains the coordinates of the points in the chain.

Note that neither the nodes nor the polygons are explicitly stored except as parts of chains.

HEAP: An implementation of a priority list (q.v.) where N elements are stored in contiguous locations in an array H ordered so that $H_i < H_{2i}$ and $H_i < H_{2i+1}$ (assuming that $2i<N$ and $2i+1<N$). Thus $H_1$ is the least element. The least element can be deleted and new elements added in $\log(N)$ time.
INVERTED PERSPECTIVE: This is a perspective projection where the viewpoint is farther away than infinity so that the lines of sight diverge as they leave the scene. Inverted perspective may also be considered as normal perspective where the farthest instead of the nearest faces are visible.

NORMAL: The normal to a face in 3-D is a line perpendicular to every line in the plane of the face. If \( \mathbf{N} \) is a normal then so is \( -\mathbf{N} \).

OBJECT: This term is a user convenience and refers to a part of the scene such as one prism that is convenient to consider as a unit.

ORIENTED FACE: this face is part of a closed polyhedron that has an inside and outside so the face's normal can be restricted to always point from the inside to the outside of the polyhedron. In addition the face's vertices can be restricted to run anticlockwise (positively) when looked at from outside.

ORTHOGONAL PROJECTION: A projection in which the viewpoint is at infinity and the lines of projection are parallel.
COMBINATORICS OF HIDDEN SURFACE ALGORITHMS
Appendix C - GLOSSARY

PERSPECTIVE PLANE: A plane through the centerpoint perpendicular to a line from the centerpoint to the viewpoint. The projected image is formed on this plane.

PERSPECTIVE PROJECTION: A method of reducing or projecting a 3-D scene to a 2-D scene. The projection has a viewpoint and a centerpoint, through which a perspective plane is extended perpendicular to the line to the viewpoint. A point in the scene is projected onto the perspective plane by drawing a line from the viewpoint through that point to intersect the plane. Since straight lines remain straight after the projection, only the vertices need be projected since the projected edges will join their projected endpoints.

PIXEL: On a raster display device such as a TV screen, a pixel is the smallest addressable location on the screen. Its properties are intensity and possibly color.

POLYGON (of a GBF): A region of the plane that is defined by certain chains in the GBF that are its borders. It need not be connected (e.g. all Hawaii may be one polygon) and may contain holes with islands in the
holes, ad infinitum.

POSITIVE ROTATION: Anticlockwise or counterclockwise.

PRIORITY LIST: A data structure that facilitates the following two operations: 1) Add an element. 2) Locate and delete the smallest element. A convenient implementation that performs these in log(N) time is the heap. (q.v.)

PRISM: A polyhedron with a top face, bottom face, and side faces. The bottom face is a general polygon that does not intersect itself. The top is congruent to and parallel to the bottom and removed from the bottom in a direction normal to the plane of the bottom, without rotation. Each side is a rectangle corresponding to an edge of the bottom and joining it to the corresponding edge of the top.

PRISM-MAP: An implementation of the prism special case algorithm described in Chapter 4.

RADIAL TRANSFORMATION: This is a transformation of the vertices of a well tesselated polyhedron wherein the distances of the vertices, from any central point are arbitrarily changed without changing the angles.
RAY: A semi-infinite line extending from a given endpoint in a given direction towards infinity.

SCALE INVARIANT CURVE: A curve whose statistical properties do not depend on how it is scaled. Natural boundaries such as shorelines may be such curves.

SCENE: The entire 3-D ensemble of objects, whether polyhedra, spheres or whatever that is the input to the hidden surface algorithm to be plotted.

SCREEN: The usually square area on which the plot is produced. This may be a CRT, page of paper in a printer, or sheet in a plotter.

SEGMENT: A short straight line between two adjacent points in a chain.

SPLINE: A technique for interpolating a smooth function through a set of N points, \((x_i, y_i)\), where \(x_i < x_{i+1}\) for \(1 \leq i < N\). The spline function between each set of consecutive points is a separate function from a certain class. The functions are joined continuously with continuous derivatives at the \((x_i, y_i)\). A popular class of functions is the class of cubic polynomials in which case the splines are cubic splines.
THREE-POINT PROJECTION: A perspective projection in which the viewpoint is not at infinity. Any scene subject to a 3-D point projection can be transformed to a corresponding scene that when subject to an orthogonal projection produces the same plot. This transformation preserves straight lines and flat planes. It makes hidden surface calculations much easier since it is simpler to determine hiddenness and do clipping with an orthogonal projection.

VERTEX: A 3-D point in the scene that delimits edges and faces.

VIEWPLOT: An implementation of the fast object space algorithm described in Chapter 3.

VIEWPOINT: The 3-D point from which the imaginary observer is looking at the scene. He is looking in the direction of the centerpoint.

WELL TESSELATED POLYHEDRON: A polyhedron where: 1) The faces are triangles. 2) There is a central point inside from which every face is totally visible. 3) For any pair of edges incident on the same vertex; if the edges are adjacent their angle is less than 90° and otherwise it is greater than 90°.
WORLD DATA BANK II (WDB-II): A GBF listing world national boundaries available from the NTIS.
APPENDIX D

BIBLIOGRAPHY

The entries are identified by the author's last name (and possibly initials or co-author) and the year (possibly suffixed by a, b, c, ...) of publication.

Aho, Hopcraft & Ullman [1974]

World Data Bank II: Content, Structure & Application. An
Advanced Study Symposium on Topological Data Structures for
Geographic Information Systems, Oct 16-21, 1977, Laboratory
for Computer Graphics and Spatial Analysis, Graduate School
of Design, Harvard University.

Appel, A. [1967]
The Notion of Quantitative Invisibility and the Machine
387-393.

Archuleta, Michael [1972]
Hidden Surface Line Drawing Algorithm. U. of Utah,
UTEC-CSc-72-121, June 1972.

Scan Conversion algorithms for a cell organised raster

Bentley, Jon Louis [1975a]
Divide & Conquer in Multidimensional Space. Symp. on the
COMBINATORICS OF HIDDEN SURFACE ALGORITHMS

Appendix D - Bibliography

Bentley, Jon Louis & Stanat, Donald F [1975b]

Blinn, James F. [1977]

Blinn, James F & Newell, Martin E [1976]

Braid, I C [1975]

Brauer, Charles R. [1968]


Brown, Christopher M. [1977]

Burton, Warren [1977]

Catmull, E. [1974]

Chrisman, Nicholas [1977]
COMBINATORICS OF HIDDEN SURFACE ALGORITHMS

Appendix D - Bibliography

Clark, James H [1976a]

Clark, James H [1976b]

Cline, A K [1974]

Coons, Steve Anson [1966]

Crow, Franklin C [1976]

Crow, Franklin C. [1977a]

Crow, Franklin C. [1977b]

Crow, Franklin C [1978]

Donath, W. E. [1968]

Douglenik, James [1977]

Douglas, David & Peucker, Tom [1973]
Algorithms for the reduction of the number of points required to represent a digitized line or its caricature.


COMBINATORICS OF HIDDEN SURFACE ALGORITHMS

Appendix D - Bibliography


Greenberg, Donald P. [1977b]

Greville, Thomas Nall Eden (ed) [1969]

Hamlin, G. & Gear, G.W. [1977]

Herman, Gabor T. & Liu, H. K. [1977a]

Herman, Gabor T. & Naparstek, Abraham [1977b]

Jarvis, John P [1975]

Jordan, B. W. & Barrett, R. C. [1973]
A Scan conversion algorithm with reduced storage requirements. Comm. ACM 16(11), Nov. 1973, pages 676-82.

Knuth, Don E [1973]

The Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University [1977]

The Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University [1978]

Levin, Joshua [1976]

Little, D. V. [1974]

Loutrel, P. P. [1967]

Loutrel, P. P. [1970]

Mahl, R [1972]

Mandelbrot, Benoit B. [1977]

Melzak, Z A [1973]

Moran, P. A. P. [1966]

Moran, P. A. P. [1969]

Myers, Allan J. [1975]
COMBINATORICS OF HIDDEN SURFACE ALGORITHMS

Appendix D - Bibliography

Negroponte, Nicholas [1977]
Raster Scan Approaches to Computer Graphics. Computers &

Newell, M E & Newell, R G & Sancha, T L [1972a]
A New Approach to the Shaded Picture Picture Problem. Proc
ACM 1972 Nat Conf.

Newell, M E [1972b]
Solution to Hidden Surface Problem. Proceedings of the ACM
1972 Annual Conference, Boston, pages 443-450.

Newell, Martin E. & Blinn, James F. [1977]
The Progression of Realism in Computer Generated Images. ACM
1977 National Conference.


Nir, Shlomo & Garduno, Ramon & Rein, Robert [1977]
Display, Manipulation, and Simulation of Macromolecules.
International J. Quantum Chem., Quantum Biology Symp. 4,

Orr, Joel N [1978]
46-71.

Parent, R.E. [1977]
A system for sculpting 3-D data. Computer Graphics 11(2),

Pavlidis, Theodosios [1976]
The Use of algorithms of piecewise approximations for
picture processing applications. TOMS 2(4), Dec. 1976,
p305-21.

Phong, Bui Tuong [1975]
Illumination for Computer Generated Pictures. Comm. ACM
18(6), June 1975, pages 311-6.

Pooch, Udo W [1976]
Computer Graphics, Interactive Techniques & Image
COMBINATORICS OF HIDDEN SURFACE ALGORITHMS

Appendix D - Bibliography

Potmesil, Michael [1976]
An Implementation of the Loutrel Hidden-Line Algorithm.

Potmesil, Michael [1977]

Rabin, Michael O [1976]
Probabilistic Algorithms. IBM RC 6164 (26545), 24 August
1976.

Rivest, Ronald L [1976]

Rivest, Ronald Linn [1974]
Analysis of Associative Retrieval Algorithms. Stanford U.,

Roberts, L. G. [1963]
Machine Perception of Three-Dimensional Solids. MIT Lincoln
Lab, Tech. Rept. 315, May 1963
(also in) Optical and Electro-Optical Information

Rogers, David P. & Adams, J. Alan [1976]
Mathematical Elements for Computer Graphics. McGraw-Hill,
1976.

Schumacker, Robert A. & Brand, B. & Gilliland, M. & Sharp,
W. [1969]
Study of applying computer generated images to visual
simulation. AFHRL-TR-69-14, U.S. Air Force Human Resources
Lab, Sept. 1969.

Shamos, Michael Ian & Hoey, Dan [1975a]
Closest point problems. Yale U., Dept. of Computer Science

Shamos, M. I. [1977b]
Geometric Complexity. ACM Symposium on the Theory of
Computing 7, May 1975.

Shamos, M. I. [1975c]
Problems in Computational complexity. Carnegie Mellon
COMBINATORICS OF HIDDEN SURFACE ALGORITHMS

Appendix D - Bibliography

University, Dept. of Computer Science, May 1975.

Shamos & Yuval, Gideon [1976a]
Lower Bounds from Complex Function Theory. Symp. on the

Shamos [1976b]
Geometric Intersection Problems. Symp. on the Foundations of
Computer Science 1976.

Skinner, Frank D. [1966]
Computer graphics - where are we?. Datamation 12(5), May
1966.

Sutherland, Ivan E [1966]
Ten Unsolved Problems in Computer Graphics. Datamation

Sutherland, Ivan E. & Sproull, Robert F & Schumacker, Robert
A [1974b]
Computing Surveys, 6(1), March 1974.

Tanimoto, S.L. [1977]
A graph-theoretic real-time visible surface editing
technique. Computer Graphics 11(2), ACM SIGGRAPH'77

Tobler, Waldo [19??]
A Computer Program to Draw Perspective Views of Geographic
Data. Cartographic Lab Report 1, Dept. of Geography, U. of
Michigan. (undated).

Veen, Arthur & Peachey, Lee D [1977]
TROTS: A computer graphics system for 3-D reconstruction
135-50.

Warnock, J E [1969]
A Hidden Line Algorithm for Half-tone Picture
Representation. U of Utah, Dept. of Computer Science Tech.
Rept. 4-15, June 1969.

Warnock, John E. [1970]
A Hidden Surface Algorithm for Computer Generated Halftone
Watkins, G S [1970]
A Real Time Visible Surface Algorithm. U of Utah, Dept. of

Watkins, Steven L. [1974]
Marked 3-D plot program with rotations. Comm. ACM 17(9),

Weiler, Kevin & Atherton, Peter [1977]
Hidden surface removal using polygon area sorting. Computer
Graphics 11(2), ACM SIGGRAPH'77 Proceedings, Summer 1977,
pages 214-228.

White, Dennis [1977]
A New Method of Polygon Overlay. An Advanced Study Symposium
on Topological Data Structures for Geographic Information
and Spatial Analysis, Graduate School of Design, Harvard
University.

Williamson, Hugh W. [1972]
pages 100-3.

Woon, P. & Freeman H [1972]
A Procedure for Generating Visible-Line Projections of
Solids Bounded by Quadric Surfaces. Information Processing

Wright, Thomas [1974]
Visible surface plotting program. Comm. ACM 17(3), March
1974, Algorithm 475, pages 152-5 Trans. on Math. Software
2(1), March 1976, pages 109-10 (Remark).

Wu, Sheng-Chuan & Abel, John F. & Greenberg, Donald P.
[1977]
An interactive computer graphics approach to surface

Yuval, Gideon [1975]
Finding Near Neighbours in K-Dimensional Space. Info Proc
Active edge ............ 4-19

Algorithms
   image space ........ 1-2, 2-20, 2-32, 7-2
   list priority ....... 2-20
   object space ........ 1-2, 2-20, 2-23, 7-2
   probabilistic ....... 7-8

Aliasing ............... 2-39

Animation ............... 2-2

B-spline ............... 2-8

Back face ............... C-1

Ball model ............... 1-8, 5-1

Bottom edge ............. 4-13

CAD ..................... 2-2

Cartography ............. 1-4, 7-2

Centerpoint ............. 2-11, C-1

Central point ........... C-1

Chain ................... 1-4, C-2

Chemistry, ............... 5-2
Clipping . . . . . . . . . . . 2-18, C-2
Coherence . . . . . . . . . . . 2-21
Computational geometry . 7-8
Computer aided design . . . 7-6
Computer axial tomography . 2-37
Constant density packing . 5-6
Convex polyhedron . . . . . C-2
Crosshatch line . . . . . . . 6-4
Cubic spline . . . . . . . . . . 2-7
Curve
    scale invariant . . . . . . C-7
Deproject . . . . . . . . . . . 3-17, 3-31, 3-33
Digitizer . . . . . . . . . . . 1-4
Dimension
    fractional . . . . . . . . . C-3
Direction cosine . . . . . . . C-2
Directly hide . . . . . . . . . 4-11
Display device . . . . . . . . 6-1
Edge . . . . . . . . . . . . . . . . 2-6, 3-3, C-2
Electron orbital clouds . . . 2-37
External sorting .......... 6-15

Face .................... 2-6, 3-3, C-2
   oriented ............... C-4
Fractional dimension curve C-3

GBF ........................ C-3
Generalization ............ 1-8
Geographic base file ..... C-3
Geography .................. 1-4, 7-2

Hamming code ............... 6-18
Heap ........................ C-4
Hide ........................ 4-10
Horizon line ................ 2-36, 4-4, 4-24

Image space algorithms .... 1-2, 2-20, 2-32, 7-2
Implementation ............. 1-3, 1-11
Indirectly hide ............ 4-11
Infinity ..................... 3-39
Inverted perspective ...... 2-17, C-4
Isometric projection ...... 2-14
K-d tree ............... 3-25

Line of sight ............. 2-11
List
  priority ............... C-6
List priority algorithms . 2-20

Molecular model ............ 5-1
Molecule, ................. 5-1

Normal .................... C-4
Normalization ............. 2-15

Object ..................... C-4
Object space algorithms .. 1-2, 2-20, 2-23, 7-2
Objects. ................... 2-5
One point projection ....... 2-14
Optical illusion ............ 7-9
Oriented face ............... C-4
Orthogonal projection ....... 2-14, 4-8, 5-12, C-4

Packing
  constant density ......... 5-6
Partial order ............... 4-10
COMBINATORICS OF HIDDEN SURFACE ALGORITHMS
Appendix E - INDEX

Patch ............... 2-7
quadric ............ 2-8
Perspective
    inverted ........... 2-17, C-4
Perspective plane .... 2-11, 5-12, C-5
Perspective projection .. 3-13, 4-8, 5-1, C-5
Perspective projection... 2-11
Picture processing .... 2-2
Pixel ................ 2-22, C-5
Plotter ............... 2-38, 6-1
Polygon .............. 1-4, C-6
Polyhedron
    convex ............ C-2
    well tessellated ... C-8
Priority list .......... C-6
Prism ................ 1-5, C-6
PRISM-MAP ............. 4-3
Probabilistic algorithms 7-8
Projection
    isometric ........ 2-14
    one point ........ 2-14
    orthogonal ..... 2-14, 4-8, 5-12, C-4
    perspective .... 4-8, C-5
three point 2-14, C-8

two point 2-14

Quad tree 3-25
QUADRAW 2-9
Quadric patch 2-8

Radial transformation 2-36, C-6
Raster device 1-10, 6-1, 7-2
Ray C-7

Scale invariant curve C-7
Scan line 6-1
Scene 2-5, 5-12, C-7
Screen C-7
Segment 1-4, C-7
Semiactive edge 4-22
Shading 1-6
Slice 4-13
Sorting 2-21
external 6-15
Sphere 5-1
Spline ................ 2-7 to 2-8, C-7
Statistical analysis .... 1-7

Three point projection .... 2-14, C-8
Tomography ............... 2-37
Top edge .................. 4-13
Transformation
    radial ................ C-6
Tree
    k-d .................... 3-25
    quad ................... 3-25
Two point projection ...... 2-14

Variable grid ............. 3-25
Vector plotter ............. 1-10, 6-1, 7-2
Vertex .................... 2-6, 3-3, C-8
VIEWPLOT .................. 1-3, C-8
Viewpoint .................. 2-11, 2-14, 4-8, 5-12, C-8

WDB-II ..................... C-9
Well tessellated polyhedron 2-35, C-8
World Data Bank II ........ C-9