

Graphics visualization of crystalline structure
by Fan Yang

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University
© Copyright by Fan Yang (1991)

Abstract:
Scientific visualization is one of the major application fields of computer graphics. Data from experiment, or from a model, are used to demonstrate the internal geometric structures of a matter. Computer graphics has been shown to be a powerful tool in compound research.

In this research project, a complete software package is developed, coded and tested to display the spatial structure of proton and alkali of a crystal by using volume rendering techniques for volumetric data. To provide a flexible and fast rendering method, in this project, several displaying schemes are developed providing the user such important features as emphasizing only one kind of particle, rotating 360 degrees, zooming into a local structure, and many other useful functions.

The whole package is in a menu-driven style and organized in a hierarchical structure by functional modules which are implemented by C. It is developed on a HP SRX-350 workstation. With simulations on 6144 elements of volumetric data, this package has been proven to work efficiently and present a good balance between the rendering speed and screen visual effect.

GRAPHICS VISUALIZATION OF CRYSTALLINE STRUCTURE

by

Fan Yang

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

June 1991

N378
Y165

APPROVAL

of a thesis submitted by

Fan Yang

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

June 11, 1991
Date

J Dunbigh Stanley
Chairperson, Graduate Committee

Approved for the Major Department

June 11, 1991
Date

J Dunbigh Stanley
Head, Major Department

Approved for the College of Graduate Studies

June 14, 1991
Date

Henry L Parsons
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Dean of Libraries when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Fan Xong

Date May 24, 1991

TABLE OF CONTENTS

Page

1. INTRODUCTION 1

2. SIMULATION MODEL 5

Volumetric Data Model 5
Slice Model 7

3. IMPLEMENTATION 9

System Diagram 9
Major Data Structure 11
Modules 12
Construct 12
Initialize Devices 13
Initialize Parameters 13
Pop Up Main Menu 13
Close Devices 13
Drawspheres 13
Drawline 15
Showlocal 16
Single Cell 17
Cell Slice 17
Cell Plane 18
Newdip 18

4. FURTHER CONSIDERATION 19

5. RESULTS AND CONCLUSIONS 21

REFERENCES CITED 23

APPENDIX 25

Appendix -- Figures 26
Volumetric Data Model 27
Slice Models 28
System Diagram 29
Source Code 30
Input Data Sample 83

LIST OF FIGURES

Figure Page

1. Volumetric Data model 27
2. Slices Models 28
3. System Diagram 29
4. Source Code 30
5. Input Data Model 83

ABSTRACT

Scientific visualization is one of the major application fields of computer graphics. Data from experiment, or from a model, are used to demonstrate the internal geometric structures of a matter. Computer graphics has been shown to be a powerful tool in compound research.

In this research project, a complete software package is developed, coded and tested to display the spatial structure of proton and alkali of a crystal by using volume rendering techniques for volumetric data. To provide a flexible and fast rendering method, in this project, several displaying schemes are developed providing the user such important features as emphasizing only one kind of particle, rotating 360 degrees, zooming into a local structure, and many other useful functions.

The whole package is in a menu-driven style and organized in a hierarchical structure by functional modules which are implemented by C. It is developed on a HP SRX-350 workstation. With simulations on 6144 elements of volumetric data, this package has been proven to work efficiently and present a good balance between the rendering speed and screen visual effect.

CHAPTER 1

INTRODUCTION

To visualize the structure of a compound has long been a dream of physicists and chemists during the course of deepening our understanding of this world. With the developments in computer science and technologies, computer graphics provides a powerful tool to show the internal relative position, rotation, and many other geometrical properties of a compound. With more and more powerful graphics workstations available, researchers in the scientific and computing communities have developed a lot of new methods for visualizing discrete multidimensional data. Volume rendering is such a technique for visualizing sampled scalars or vector fields of the three spatial dimensions without fitting geometric primitives to the data [1]. Since all voxels participate in the generation of each image, rendering time grows linearly with the size of the data set and the complexity of the simulation model for each voxel.

The purpose of this thesis project is to render the three spatial dimensional crystal structure of proton and alkali using volume rendering technique. The 3-D structure of the volumetric data is displayed totally or zoomed locally and many choices of demonstration models are provided to achieve

better screen effect.

In inorganic chemistry, to analyze any kind of compound, two important aspects need to be considered. One is the composition of the compound. Any compound is composed of many groups. The groups of the compound always show the chemical property of the compound. Usually, for analyzing the composition of the compound, x-ray techniques are used. When the x-ray strikes on different groups, the distinguished spectrum can be obtained. This basic concept creates a method to know the composition of the compound.

Another important aspect is the geometric structure of the compound. In inorganic chemistry, it is called the crystal structure of the compound for any solid compound. The crystal structure of the compound can clearly display the force or energy between the particles and tell the chemical property of interaction. Therefore, it is also very important to show the geometric structure.

Unfortunately, the crystal structure of the compound could not be displayed in a traditional way. In the past, it was impossible to see the structure, because a compound usually consists of polymolecules. There was no efficient way to demonstrate even a small part of the structure which has hundreds of particles.

Nowadays, computer graphics is matching this problem. Three dimensional computer graphics gives a good description of the crystal structure of compound. From prepared data which

are derived from experiments, for example x-ray, three dimensional crystal structures of compound can be displayed on the screen. This project is motivated by this demand and renders the 3-D crystal structure of compound which consists of four particles: positive and negative protons and alkalis.

In crystallography, eight points of a crystal lattice defines a unit cell. The eight points are the corners of the cell and must be chosen so that they define a box having pairs of parallel faces. Every cell in a crystal is packed with atoms of exactly the same kinds, in the same numbers, and in the same relative positions. Unit cells and crystal lattices, which are products of the human imagination, allow us to visualize a crystal's structure as the result of a tidy stacking up (side by side, front to back, and bottom to top) of little bricks. A unit cell has 12 edges, 8 corners, and 6 faces. The size and shape of a unit cell are fully described by three edge-lengths and by three angles between these edges. The symmetry defines seven different shapes of the unit cell. They are: Cubic, Tetragonal, Orthorhombic, Trigonal, Hexagonal, Monoclinic and Triclinic [6]. In this project, the cubic shape is used, which has equal lengths of edges. The angles between the edges are all 90 degrees. The unit cell here has the similar expression with the cell in computer graphics used below. There are 12 equal lengths of edges, 6 parallel faces and all angles are 90 degree. The details will be discussed in Volumetric Data Model in Chapter 2.

This thesis is organized as follows, a detailed description of the model is provided in Chapter 2. The process of translating the model into an executable system is explained in Chapter 3. Chapter 4 covers some further consideration to improve the demonstration model. Finally, Chapter 5 discusses results and conclusions.

CHAPTER 2

SIMULATION MODEL

Three-dimensional voxel based graphics also refereed to as volumetric graphics or volumetric imaging, have recently emerged as a key research and development area of computer graphics. However, the rendering algorithms usually have to use too much processing time and/or memory to be implemented interactively on existing single processor workstations. This problem is only compounded as data sets increase in size and rendering algorithms increase in complexity [2].

For this project, since the number of sampled scalars can be as large as 6144, therefore, some strategies are needed to deal with the rendering speed.

Volumetric Data Model

A two dimensional 12 * 512 array of data samples forms a cube, with 8 cells on each side of the cube. All of the cells have the same size and divide the cube into 512 partitions uniformly, as shown in Figure 1. In each cell, there are 12 voxels distributed in the space of the cell according to the shift arrays given in the program. These shift arrays have lengths of 12 and are named as xshift, yshift and zshift. They will be described in detail in the section Major Data

Structure of Chapter 3. The voxel, which is the 3-D counterpart of the pixel, has a numerical value of density in the broad sense, representing the material, color, texture and transparency ratio of a small unit cube [3]. Here voxels are used to represent the particles. It includes information about the classification, the position and the simulation model of the particles.

The main object here is to display the geometric positions of particles in three dimension space. The voxels are enclosed in a cube and distributed in each predefined cell. From the array of input volumetric data, in which each cell has a designated voxel, the voxel should be decided. That is, each line that has 12 elements in the input volumetric data corresponds to a cell. The total of 512 lines belongs to the 8 * 8 * 8 (512) cells according to the values of x, y and z coordinates in three dimensions, from 0, 1, ..., 7, respectively. The order of rendering is from the lower left corner to the upper left corner, then from the back left to the back right and from the left to right. Each cell encloses 12 voxels which come from the same line in the input data array corresponding to that cell. The first 8 elements in each line define protons, either proton (1) or proton (-1), and the last four elements represent alkalis. They are alkali (1) and alkali (-1). When the volumetric data are loaded, the position of each line in the cube is decided and the relative position of each voxel in the cell is also defined by the shift arrays.

Slice Model

In many instances, regardless of the data and corresponding rendering algorithm employed, user controlled animation or motion of the data leads to significantly faster and higher understanding of the true nature of the data itself [2]. Mark Miller described two interactive programs for analyzing 3-D volumetric data in his "Final Report" [5]. These are Cubetool and Slicetool. Both operate commands are entered by a mouse. Each allows the user to observe a plane of the data volume. Cubetool allows the user to view only planes orthogonal to the primary x, y and z axes, but displays all simultaneously in orthographic and normal views. Slicetool is more general in that it allows a user to interactively analyze a volume of data by slicing the volume along arbitrary planes specified with a mouse and display the resulting data in normal view.

In this project, a similar method is used to allow the user to interactively analyze a volume of data. It provides flexible menu choices. From the menus, the x, y and z coordinates can be chosen arbitrarily by using a mouse, then a specified cell is isolated. There are many different rendering models for voxels which can be chosen from the menus with the mouse. As in Slicetool, it allows the user to specify the arbitrary slice or plane. The menus give many combination choices: a arbitrary slice of a volume of data can be displayed along x axis, y axis or z axis depending on its

(y,z) coordinates, (x,z) coordinates or (x,y) coordinates, respectively.

A more powerful function may also slice a plane by specifying the x coordinate, y coordinate or z coordinate corresponding to the y-z plane, x-z plane or x-y plane. From this point, the voxels of data here can be flexibly sliced in the 3-D space according to the user's requirement. Notice that, the unit of the slice is always the cell. Therefore, the 8 * 8 * 8 cells in this cube can be sliced in three different shapes, as shown in Figure 2 to display the local structure of the model.

The border is drawn for each slice model to show the outline of the shape. At the same time, a big cube which encloses 8 * 8 * 8 cells is displayed to provide a reference. Thus, the input volumetric data is displayed in three dimensional space by the models described above to demonstrate the crystal structure of proton and alkali.

CHAPTER 3

IMPLEMENTATION

In this part, the implementation method of the program structure and some improvement made to speed up the rendering will be discussed.

The entire program is divided into many logical modules according to the tasks. The system diagram will be discussed first. Some data structures are common to all the modules. These data structures will be briefly described before describing the functions of the various modules.

System Diagram

The system diagram is shown in Figure 3. There are three hierarchical menus in this program. They provide functions to demo whole areas, local areas and display models, respectively. The block Construct's duty is the initialization and organization of the entire program from the main menu. The main menu is prompted in the first level which gives nine choices to display the whole structures of the volume data by six different styles.

In the second level, there are nine blocks: Whole Sphere, Proton Sphere, Alkali Sphere, Whole Line, Proton Line, Alkali Line, Show local, Newdip and Exit. The block Whole Sphere

renders all voxels (6144) as spheres in a cube. It uses colors to distinguish the various particles. Proton Sphere is a special case of Whole Sphere, which shows all voxels of protons as spheres but those of alkalis as marks. It obviously emphasizes protons. A similar method is utilized in the block Alkali Sphere. As in Proton Sphere, voxels of alkalis are rendered as spheres and those of protons are displayed as marks. While these blocks demonstrate the voxels independently in a cube, WHOLE LINE uses a different idea. It shows the relationship among particles by using colored lines to connect them. Proton Line promotes the connection among voxels of protons while Alkali Line links voxels of alkalis. They all use colored lines.

The block Newdip and Show Local prompt the next menus. They are shown in the third level of the structure in this diagram. The local structures are represented by two forms. Each of them gives three rendering choices. The selections of cube or slice make the local demonstration flexible. The symmetric structure is easy to implement in the program.

The third menu level pops up after x, y, z axes are determined. From the Newdip branch, there are two display models: one is the New Cube, which uses small cubes to render the voxels, another one is the New Pyramid, which uses pyramids instead of the small cubes. In the Show Local branch, the two models are Local Line and Local Sphere. They have similar meaning with the Whole Line and Whole Sphere

blocks in the second level. The only difference is the former represents local structures chosen. The details will be described in Modules in Chapter 3.

Major Data Structure

The prot and alka are 8 * 8 * 8 * 8 and 8 * 8 * 8 * 4 matrixes, respectively, which represent voxels that keep input data. The values of input data for both matrixes prot and alka are either -1 or 1, which represent four different sorts of particles, that is proton (1), proton (-1), alkali (1) and alkali (-1). Therefore, four colors are used to represent these four particles. In prot[8][8][8][8] and alka[8][8][8][4], the first three indexes correspond to the 3-D coordinates of each cell. The last index refers to the position of the voxel in each cell. The relative position of the voxel in the cell is predefined by its chemical property. It distributes the voxels uniformly in the cell according to their physical definition. Therefore, arrays of fixed position shift are used for each 12 voxels in every cell. They are named as xshift, yshift and zshift. The three data structures are arrays with length of 12. Since each voxel has x, y, and z coordinates in the program, they are shifted according to xshift, yshift and zshift, respectively. As described before, the values of prot and alka (1, -1) indicate the four kinds of particles by color purple, blue, yellow and green in the program.

Modules

In this section, the function of each module will be discussed.

Construct

Construct is the main program. It implements the organization of the entire program by given hierarchical menu choices. It is also in charge of initialization and data loading by calling the subroutines included in module Tools.

The module Tools consists of several subroutines used as tools in the program. LoadData is used to input the data to the matrixes prot and alka. The initialization is done in the subroutine Init. First, it extends the coordinates along the direction of x, y, z in both Mouse and Overlay planes by calling subroutine Extend. Then it sets the necessary parameters, such as curve_resolution, light source, shade_mode, backface control, interior_style, vertex_format, surface_model, camera reference (the direction of x, y, z), camera lens angle, projection,etc. All of these parameters decide the effect of the graphics on the screen.

The curve_resolution is very critical here. There is a tradeoff between the resolution of sphere and the speed of rendering. The higher the resolution chosen, the better the shape of the sphere, but the rendering speed will be lower. Here, the reasonable parameters are chosen from many experiments. Camera.field_of_view decides how close the image

is seen.

The OpenDev, CloseDev and ClearScreen are used in the Construct to manage the devices. The FindChoice provides the main menu for Construct.

The Construct directs the program by following these steps:

Initialize Devices. There are three input/output primitives used: Mouse, Overlay and Display. The Mouse is the input primitive which is used in the menu selection. Overlay and Display are output primitives. Overlay is for menus, and Display is for the images.

Initialize Parameters. As described above, the Init in module Tools is in charge of this.

Pop Up Main Menu. The main menu whose choices are shown in the FindChoice pops up here. It will continue to bring up the menus hierarchically from this step and go to the different demonstration layers.

Close Devices. Once Exit in the main menu is hit, the program closes all the opened devices (Mouse, Overlay and Display) and ends execution.

Drawspheres

The function of this module is to render the 3-D crystal structure of proton and alkali by spheres using four colors. It shows the whole structure. That is 8 * 8 * 8 cells and in

each cell, there are 12 voxels rendered as spheres. The border between each cell is not displayed because of the density.

The whole-sphere, proton-sphere and alkali-sphere are in this module. The whole-sphere renders voxels of both proton and alkali as spheres in the same size but different colors. The colors tell the types of particles. They refer to the proton (1), proton (-1), alkali (1), and alkali (-1), respectively. In the program, the sum of the voxel value and a constant are used to define the index of color.

The proton-sphere and the alkali-sphere provide the ability to emphasze the structure of either protons or alkalis respectively. For example, the proton-sphere renders protons as sphere and draws marks for alkalis. It is similar to the whole-sphere. The slight difference is that it only uses two kinds of operations for protons and alkalis, so that one of them is shown more clearly on the screen. Here in the proton-sphere the proton is promoted according to its size and rendering model. The alkali-sphere is an alternative of the proton-sphere. It promotes the alkali and draws marks for protons. This time, the structure of alkali can be seen more clearly. Both rendering speeds of the proton-sphere and alkali-sphere are much faster than that of whole-sphere because drawing a mark needs less time than is used for sphere.

The whole-sphere, proton-sphere and alkali-sphere provide the whole structure of the model. They are very useful to see

the macro composition. The use of backface control and projection and shading techniques increases the visual effect of demonstration. The major drawback of this model is the rendering speed is low because of the large number of voxels (6144). It is unturntable since the rendering speed is intolerably low. Therefore, another method is used in module Drawline to achieve higher rendering speed.

Drawline

The function of this module is using line to connect the particles. In each cell the same color line is used to connect the particles of the same category. Using distinguished colors shows the connections among different kinds of particles. Since there are 12 voxels in a cell, the colored lines show their relationship among these 12 voxels. One kind of voxel has lines linked with the same kind of voxels in the cell by the same color. After experimenting, it is about 5 times faster than rendering spheres. Therefore, making it turn 360 degrees is possible.

In order to make the image turntable, the main idea is using the double buffer. The double buffer implements the display frame by frame alternately. On the screen, a smooth turntable image is shown. The requirement to achieve this effect is the fast rendering speed. The quicker the rendering, the smoother the image turns. Another useful strategy used to make the image turntable is adjusting the camera position instead of turning the image around. Turning

the camera in a certain direction gives the impression that the image is turning in the opposite direction.

Two functions detect-prot and detect-alka are provided to detect the 12 voxels in each cell. For any voxel in a cell, detect-prot is used to detect a proton among its undetected neighbors, where the number of neighbors is 11, 10, 9, ..., 2, 1. After an identical type of proton is detected, a colored line is drawn between these two voxels. The detect-prot includes detection for both proton (1) and proton (-1). The same idea is used in detect-alka for detecting alkali (1) and alkali (-1).

In this model, there are three subroutines. All the links occur among the 12 voxels in a cell for these three subroutines as described above. The whole-line uses detect-prot and detect-alka to test the neighbors in each cell. It links all the voxels which have the relationship as described above, respectively, including both proton (1, -1) and alkali (1,-1). Therefore, proton-line and alkali-line are the special versions of the subroutine whole-line. Because of the decrease of the number of voxels to detect each time, proton-line and alkali-line are much faster than the whole-line.

Showlocal

This module functions by zooming the local structure in a cell as a unit. As described in the System Diagram, the module of Showlocal provides two layers of menus. The first layer is the position selection. These menus will pop up for

choosing x, y, and z coordinates, respectively. Then in the next layer, there are two subroutines: local-sphere and local-line. The local-sphere uses sphere as the rendering model for each voxel. Four colors are used here. The local-line has the similar function as in Drawline. The difference is that the first shows the local structure as indicated. There are two functions in this module: one is draw-cubic which shows the outline of the cells; another is DrawSmCubic. The latter provides a small cubic reference for the outline around the border of the cells.

The other subroutines in this module provide multiple ways to zoom in the local structure in four different ways. They are Single Cell, Cell Slice and Cell Plane:

Single Cell. Render 12 voxels within any single cell as sphere model or linked line in one of the 8 * 8 * 8 cells. The value of the voxel corresponds to the color of a specific particle, proton (1), proton (-1), alkali (1) and alkali (-1). The small cube around the border of each cell gives the reference for the configuration as in Figure 3. When it turns around, the relative position in 3-D space will be shown.

Cell Slice. Render 12 * 8 voxels within an 8 cell slice as sphere model or linked line in 8 * 8 * 8 cells along either x, y or z axis. The small rectangle slice also provides the reference of the position in 3-D space. The structure is shown in the slice unit.

Cell Plane. Render 12 * 8 * 8 voxels within an 8 * 8 cell plane as sphere model or linked line in 8 * 8 * 8 cells along x-y plane, y-z plane or z-x plane. It demonstrates the plane of the rectangle in 3-D space. Its unit is the cell. It is also turntable.

Because Showlocal renders a structure locally, it takes much less time, but it can show the model clearly.

Newdip

The module Newdip is a different version of the module Showlocal which improves the rendering speed. It has the same functions to demonstrate the local structure of the model and is designed to demonstrate the 360-degree turntable image. The difference is that Newdip consists of two main subroutines: New-cube and New-pyramid. Both of them have the same duties as those of local-sphere, but use different rendering models for the voxels. New-cube uses small cube as rendering model. It is much faster than local-sphere, since local-sphere needs more time to draw each sphere. New-pyramid makes the pyramid as its rendering model. It is easy to show that the speed of rendering New-pyramid is three times quicker than that of New-cube because the ratio of the line numbers from cube to pyramid is 3:1.

These major modules described above form the entire demonstration program. The different ways to render are shown using choices in the hierarchical menu system. Therefore, the user interface is flexible and convenient.

CHAPTER 4

FURTHER CONSIDERATION

Currently this program has been implemented for rendering volume data in a uniformed cube in 3-D space. It rotates in a certain direction which is a 360 degree rotation from the right to the left around the center of the cube. However, some other options can be provided to enrich the demonstration further.

One choice is to increase the flexibility of the rotation. For example a new procedure Rotate-Choice may be created. The procedure Rotate-Choice provides a new menu so that the rotation direction, the rotation center and the rotation degree can be decided by users as they like. This will not significantly increase the complexity of the program and it is also a minor job.

Another improvement is to create a new function Zoom. Zoom shows the ability to focus on the area as indicated and enlarges or shrinks the structure. To do this, a procedure can be used to set the length of camera according to the requirement so that the length of camera is variable. Instead of rendering fixed size of cells each time, Zoom may provide choices of the size of cell and model of each voxel. This improvement obviously fancies this program.

A quite different idea comes from the simulation of the model of the electron density clouds in a covalent bond [4]. The main idea is to show the distribution of the same kind of particles by rendering a surface contour. Because of the large number of voxels, it may be divided into several layers. It is convenient to create eight layers. In each layer, the curved surfaces are rendered independently on one kind of particle. The colors of the surfaces can still be used to show the different particles. To implement this idea, much more work needs to deal with the surface rendering predictably. The time complexity will be greatly increased. Therefore, the interactive rendering of the large volume of data may run into difficulties.

CHAPTER 5

RESULTS AND CONCLUSIONS

As discussed earlier, the efforts in this project have focused on the simulation and rendering models based on the rendering speed for interactive analysis. The simulation was carried out on 12 * 512 (6144) elements of volumetric data, which define the positions in the 3-D space and the kinds of particles (positive and negative protons and alkalis).

The simulation was displayed on a HP SRX-350 work station. The Starbase graphics package is used in design. The entire program is written in C. The System Diagram is listed in Figure 3 in the Appendix. All the programs listed in Figure 4 are in the Appendix and a group of input data sample needed for in Figure 5 is in the Appendix B.

As can be expected, the larger the volume of the data set, the longer the rendering time required. Furthermore, the more complicated the rendering model, the longer time will be needed for the same volume of data. This can be shown by comparing the case when using the sphere model with the case when using the cube or pyramid models in voxel rendering. It takes much more time to render a sphere than to render a cube or pyramid. The increased amount of rendering time from pyramid to cube is linear.

Since many conflicts are encountered, such as rendering speed versus image resolution, speed versus displayed detail, etc, many demonstration methods have been provided in this program to make a good trade off. The lines, marks, colors and so on are proposed to achieve higher rendering speed at the same time without rendering the visual effect of final display by too great a degree. The flexible choices obviously increase overall efficiency to display the structures. The crystal structure of volumetric data of proton (1, -1) and alkali (1, -1) can clearly be observed in many demonstration models. The goal of this project has been achieved. This program provides a demonstration method for the crystal structure of polymolecules compound in inorganic chemistry. All the programs have been tested and they can efficiently be used to render the 3-D structure of a compound. Therefore, with some more developments, results from this project will make the demonstration of the crystal structure of compound completely possible.

REFERENCES CITED

REFERENCES

[1] M.Leovy, Efficient Ray Tracing of Volume Data. ACM Trans. Graphics, Vol. 9, Number 3, July 1990, pp.245-261

[2] K.I.Joy, Parallel Algorithmic Methods for Volumetric Rendering. GRAI9023, ERA9044, Feb. 1990, Dept. of Energy, Washington, D.C.

[3] M.Levoy, Display of Surfaces from Volume Data. IEEE comput. Graph. Appl. 3, 8, May 1988, pp.39-47

[4] J.Blinn, A Generalization of Algebraic Surface Drawing. ACM Trans. Graphics, Vol. 1, Number 3, July 1982, pp.235-256

[5] M.C.Miller, Pixar Data Visualization Tools Overview --- Final Report. 068147000, 9513035, Feb. 1990, pp.23, Dept. of Energy, Washington, D.C.

[6] D.W.Oxtoby, N.H.Nachtrieb, W.A.Frecman, Chemistry:Science of Change. Saunders College Publishing, a Division of Holt, Rinehant and Winston, Inc., 1990, pp.815-853

[7] C.K.Pokorny, C.F.Gerald, Computer Graphics: the Principles Behind the Art and Science. Franklin, Beedle & Associates, 1989

[8] Hewlett-Packard Company, Starbase Graphics Techniques: HP-UX Concepts and Tutorials. Volume I, II

APPENDICES

APPENDIX A

FIGURES

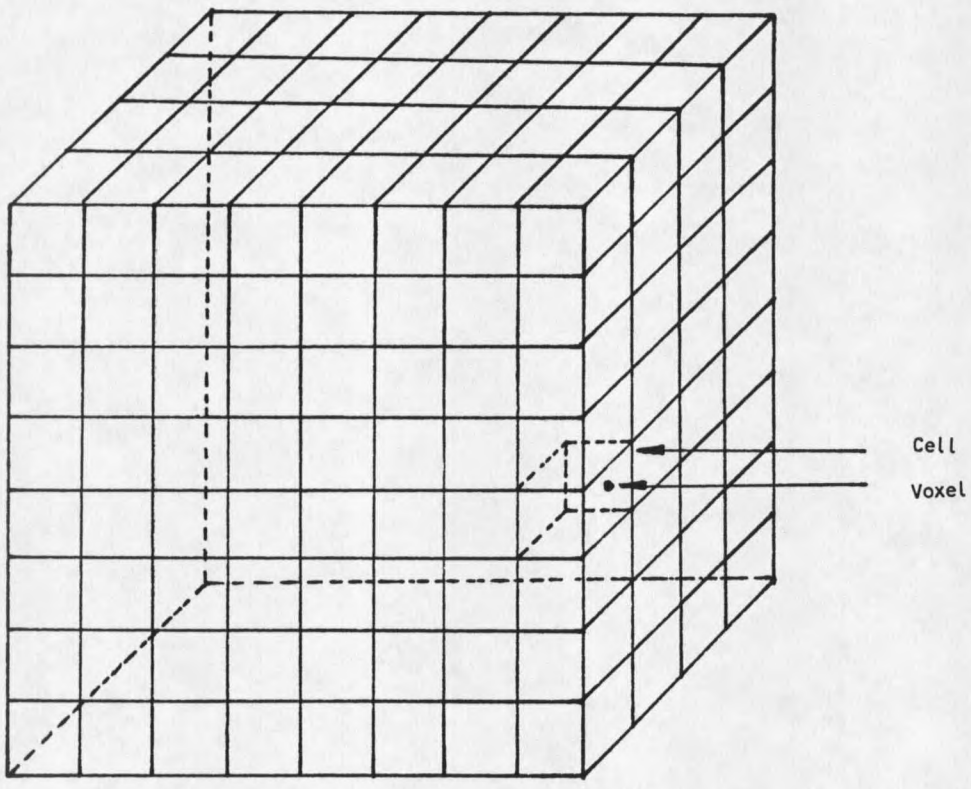


Figure 1. Volumetric Data Model

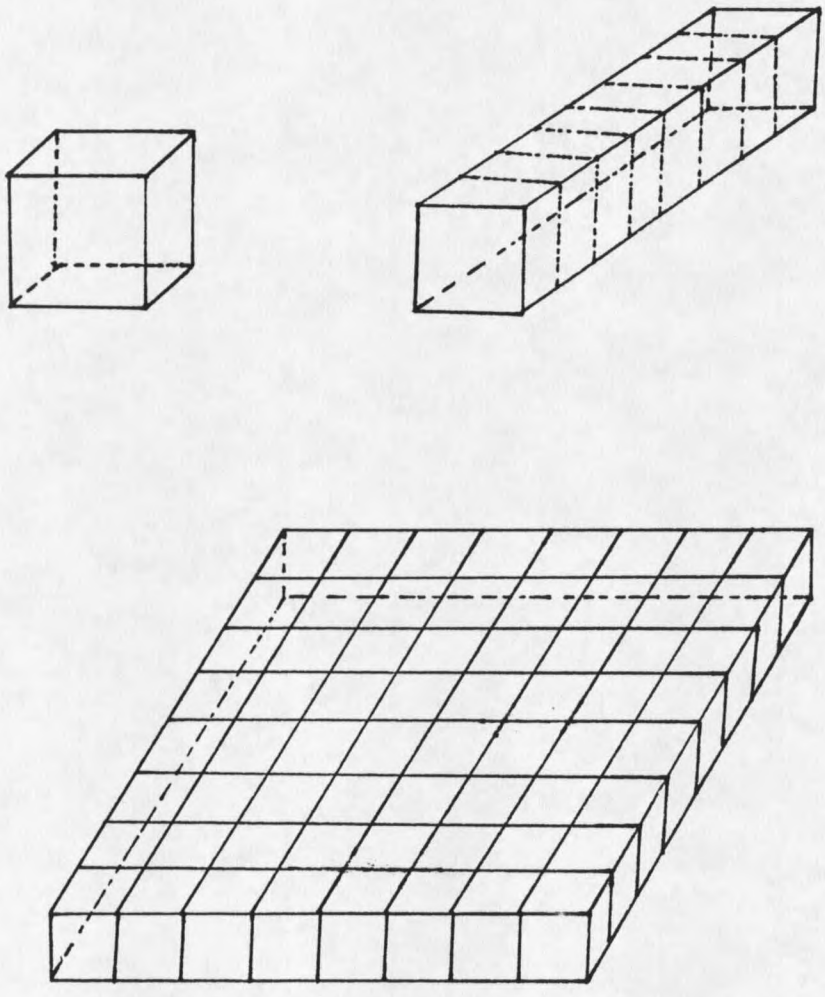


Figure 2. Slice Models

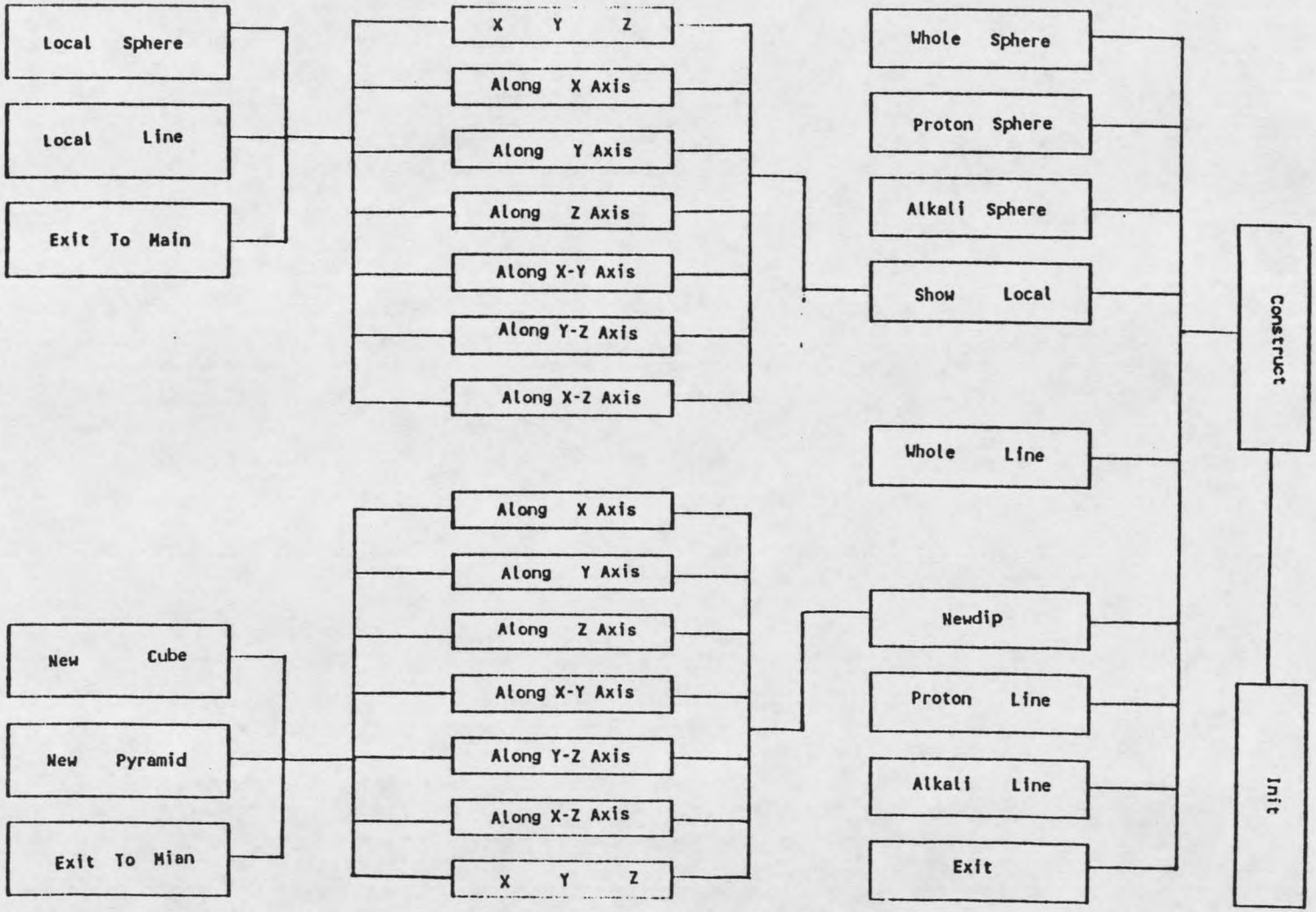


Figure 3. System Diagram

Figure 4 · SOURCE CODE

```
/*************************************************************************/
/*   File Name:   construct.c                    Author: Fan Yang        */
/*                                                                       */
/*   Files used with this package:   construct.c                        */
/*                                   drawsphere.c                        */
/*                                   drawline.c                          */
/*                                   showlocal.c                         */
/*                                   newdip.c                            */
/*                                   spheres.c                           */
/*                                   tools.c                             */
/*                                                                       */
/*   Functions in this file:   None                                      */
/*                                                                       */
/*   Note:   All the menu subroutines are written Chance Younkin         */
/*************************************************************************/

#include <starbase.c.h>
#include <stdio.h>

#include "whole.h"

main()
{
   void    Init(), DelAllMesg();
   void    PutMenu(),ShowChoice(),CloseDev();
   MenuSt  LoadMenu(), *GetChoice();
   int     OpenDev();
   Boolean FindChoice();
   int     Choice, OldChoice = NO_CHOICE;
   char    MenuFileName[80];
   Boolean MenuGone;
   MenuSt  MainMenu;

   Mouse = OpenDev("/dev/locator", INDEV, "hp-hil", INIT);
   Overlay = OpenDev("/dev/ocrt",OUTDEV, "hp98720", INIT);
   Display = OpenDev("/dev/crt", OUTDEV,"hp98721",INIT|THREE_D|MODEL_XFORM);
   Init();
   ClearScreen(Overlay);
   strcpy(MenuFileName, "fan.menu");
   GET_PATH(MENU_PATH, MenuFileName);
   MainMenu = LoadMenu(MenuFileName);
   PutMenu(Overlay, NULL_STRING, RIGHT_EDGE - MainMenu.Width, 0, &MainMenu);
   do
   {
      GetChoice(Mouse, Overlay, &Choice);
      ShowChoice(Overlay, Choice, OldChoice,  &MainMenu);
      DelAllMesg(Overlay);
      OldChoice = Choice;
      MenuGone = FindChoice(Choice);
      if (MenuGone)
      {
         PutMenu(Overlay, NULL_STRING, RIGHT_EDGE-MainMenu.Width, 0,&MainMenu);
         ShowChoice(Overlay, Choice, NO_CHOICE, &MainMenu);
      }
    } while (Choice != MainMenu.NbrEnts);
    CloseDev(Display);
    CloseDev(Overlay);
    CloseDev(Mouse);
}
```

```
/*************************************************************************/
/*   File Name:   tools.c                        Author:   Fan Yang      */
/*                                                                       */
/*   File used with this package:   tools.c                              */
/*                                  construct.c                          */
/*                                  drawline.c                           */
/*                                  drawsphere.c                         */
/*                                  spheres.c                            */
/*                                  showlocal.c                          */
/*                                  newdip.c                             */
/*                                                                       */
/*   Functions in this file:   LoadData()                                */
/*                             Init()                                    */
/*                             Extents()                                 */
/*                             FindChoice()                              */
/*                             OpenDev()                                 */
/*                             CloseDev()                                */
/*                             ClearScreen()                             */
/*************************************************************************/

#include <stdio.h>
#include <starbase.c.h>
#include <fcntl.h>
#include  <string.h>

#include "whole.h"

static int  Devs[10], NbrDevs = 0, Count;
void ResetView();

/*************************************************************************/
/*   Function Name:   LoadData()                                         */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  None                                                       */
/*                                                                       */
/*   Retrun:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function reads all the input data and load the data         */
/*   into the matrix prot[8][8][8][8] and alka[8][8][8][4]               */
/*************************************************************************/

void LoadData()
{
   int x, y, z;
   for (x = 0; x < 8; x++) {
       for (y = 0; y < 8; y++) {
           for (z = 0; z < 8; z++) {
               scanf (" %d %d %d %d %d %d %d %d    %d %d %d %d",
                     &prot[x][y][z][0], &prot[x][y][z][1], &prot[x][y][z][2],
                     &prot[x][y][z][3], &prot[x][y][z][4], &prot[x][y][z][5],
                     &prot[x][y][z][6], &prot[x][y][z][7], &alka[x][y][z][0],
                     &alka[x][y][z][1], &alka[x][y][z][2], &alka[x][y][z][3]);
           }
       }
    }
    return;
}

/*************************************************************************/
/*   Function Name:   Init()                                             */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
```

```
/*   Output:  None                                                          */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                          */
/*      This function initializes all the parameters used in the graphics */
/*   and call subroutine Extents to extend the coordinates of the system  */
/*   and load data.                                                        */
/****************************************************************************/

void Init()
{
   void         MakeCursor();
   void         Extents();
   gescape_arg  arg1;
   MenuSt       LoadMenu();
   char         MenuFileName[80];

   Extents ();
   arg1.i[0] = 0;
   LoadData();
   gescape (Overlay, R_TRANSPARENCY_INDEX, &arg1, &arg1);
   curve_resolution (Display, STEP_SIZE, 0.5 , 0.5, 0.5, 0.5);
   shade_mode (Display, CMAP_FULL | INIT, TRUE);
   light_ambient (Display, 0.5, 0.5, 0.5);
   light_source (Display, 1, POSITIONAL, 0.750,0.750,0.750,0.0,7.0,-5.0);
   light_source (Display, 2, POSITIONAL, 0.550,0.550,0.550,3.0,3.0,3.0);
   light_switch (Display, 7);
   interior_style (Display, INT_SOLID, NoEdges);
   backface_control (Display, FALSE, TRUE, 1.0, 0.0, 0.0);
   vertex_format (Display, 3, 3, FALSE, FALSE, CLOCKWISE);
   surface_model (Display, TRUE, 50, 1.0, 1.0, 1.0, 1.0);
   Camera.upx = 0.0, Camera.upy = 1.0, Camera.upz = 0.0;
   Camera.front = Camera.back = 0.0;
   Camera.refx = 0.5, Camera.refy = 0.5, Camera.refz = 0.5;
   Camera.projection = CAM_PERSPECTIVE;
   Camera.field_of_view = 25;
   MakeCursor (Overlay, Mouse, USER_DEF, XOR, 10.0, 8.0, 0.0, "arrow");
   return;
}

/****************************************************************************/
/*   Function Name: Extents()                                               */
/*                                                                          */
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  None                                                          */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                          */
/*      This function extends the coordinates in both Mouse and Overlay    */
/*   planes.                                                               */
/****************************************************************************/

void Extents()
{
   vdc_extent(Mouse, lox, loy, loz, hix, hiy, hiz);
   vdc_extent(Overlay, lox, loy, loz, hix, hiy, hiz);
   make_picture_current(Display);
   return;
}

/****************************************************************************/
/*   Function Name:   FindChoice()                                          */
/*                                                                          */
```

```
/*   Input:   Choice - The choice made in a menu                          */
/*                                                                        */
/*   Output:  None                                                        */
/*                                                                        */
/*   Return:  Gone - A boolean variable indicates the success of the      */
/*                   switch                                               */
/*   Description:                                                         */
/*      This function provides the switch to different subroutines by     */
/*   the value of Choice from the mouse.                                  */
/************************************************************************/

Boolean FindChoice(Choice)
int Choice;
{
   void     whole_line(), proton_line(), alkala_line();
   void     whole_sphere(), proton_sphere(), alkala_sphere();
   void     showlocal(), newdip();
   Boolean  Gone = FALSE;

   switch (Choice)
   {
      case 1:Gone = TRUE; newdip();         break;
      case 2:Gone = TRUE; whole_line();     break;
      case 3:Gone = TRUE; proton_line();    break;
      case 4:Gone = TRUE; alkala_line();    break;
      case 5:Gone = TRUE; whole_sphere();   break;
      case 6:Gone = TRUE; proton_sphere();  break;
      case 7:Gone = TRUE; alkala_sphere();  break;
      case 8:Gone = TRUE; showlocal();      break;
      case 9:break;
   }
   return(Gone);
}

/************************************************************************/
/*   Function Name:   OpenDev                                           */
/*                                                                      */
/*   Input:   Path - A path shows the device                            */
/*            Kind - The kind of input or output device                 */
/*            Driver - The driver's name                                */
/*            Mode - A mode of the device                               */
/*                                                                      */
/*   Output:  None                                                      */
/*                                                                      */
/*   Return:  Devs - A pointer to the device                            */
/*                                                                      */
/*   Description:                                                       */
/*      This function opens the device as indicated.                    */
/************************************************************************/

int OpenDev(Path, Kind, Driver, Mode)
char  *Path, *Driver;
int   Kind, Mode;
{
   if ( (Devs[NbrDevs] = gopen(Path, Kind, Driver, Mode)) == -1 )
   {
      for ( Count = 0; Count < NbrDevs; Count++ )
         gclose(Devs[Count]);
      EXIT(stderr, "Couldn't open the graphics device");
   }
   return(Devs[NbrDevs++]);
}

/************************************************************************/
/*   Function Name:   CloseDev()                                        */
/*                                                                      */
```

```
/*   Input:   Fildes - A file descriptor                                    */
/*                                                                          */
/*   Output:  None                                                          */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function closes all the devices opened.                        */
/****************************************************************************/

void CloseDev(Fildes)
int Fildes;
{
   void        ClearScreen();
   float       Rev;
   int         Type;
   char        Model[20];

   inquire_id(Fildes, &Rev, Model, &Type);
   if ( strcmp(Model, "hp98720") == 0 )
   {
      background_color_index(Fildes, 0);
      ClearScreen(Fildes);
   }
   else if ( strcmp(Model, "hp98721") == 0 )
   {
      background_color_index(Fildes, 0);
      ClearScreen(Fildes);
   }
   else if ( strcmp(Model, "hp-hil") == 0 )
      track_off(Fildes);
   gclose(Fildes);
   for ( Count = 0; Fildes != Devs[Count]; Count++ );
   Devs[Count] = Devs[--NbrDevs];
   return;
}

/****************************************************************************/
/*   Function Name:   ClearScreen()                                         */
/*                                                                          */
/*   Input:   Fildes - A file descriptor                                    */
/*                                                                          */
/*   Output:  None                                                          */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function clear the view surface on the screen.                 */
/****************************************************************************/

void ClearScreen(Fildes)
int  Fildes;
{
   clear_view_surface(Fildes);
   make_picture_current(Fildes);
   return;
}
```

```
/*************************************************************************/
/*   File Name:   drawline.c                    Author: Fan Yang         */
/*                                                                       */
/*   Files used with this package:   drawline.c                          */
/*                                   construct.c                         */
/*                                   drawsphere.c                        */
/*                                   spheres.c                           */
/*                                   tools.c                             */
/*                                   showlocal.c                         */
/*                                   newdip.c                            */
/*                                                                       */
/*   Functions in this file:   whole_line()                              */
/*                             detect_prot()                             */
/*                             detect_alka()                             */
/*                             proton_line()                             */
/*                             alkala_line()                             */
/*************************************************************************/
#include <starbase.c.h>
#include <stdio.h>
#include <math.h>

#include "whole.h"

/*************************************************************************/
/*   Function Name:   whole_line()                                       */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Whole line structure on the screen                         */
/*                                                                       */
/*   Retrun:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function draws a linked line structure on the screen and    */
/*   shows the whole structure of the model.                             */
/*************************************************************************/

void whole_line()
{
        int     x, y, z, i, Buffer=0;

         ClearScreen(Display);

         hidden_surface(Display, HSRoff, CULLoff);
         double_buffer(Display, TRUE, 8);

          for (Theta = 0.0; Theta < 360.1; Theta += 5.0) {
             Camera.camx = 5 * cos((Theta+70) deg);
             Camera.camz = 3.0;
             Camera.camy = 5   * sin((Theta+70) deg);

                view_camera(Display, &Camera);

                dbuffer_switch(Display, Buffer = !Buffer);

                for (x = 0; x < 8; x++) {
                   xrefpt = (float)x / 8.0;
                   for (y =0; y < 8; y++) {
                      yrefpt = (float)y / 8.0;
                      for (z =0; z < 8; z++) {
                         zrefpt = (float)z /8.0;
                         for (i = 0; i < 8; i++) {
                            CaseArg = prot[x][y][z][i] + 2;
                            switch (CaseArg) {
                               case 1:
          detect_prot (ProtNegCol,CaseArg,x,y,z,i,xrefpt,yrefpt,zrefpt);
```

```
                                    break;
                                  case 3:
                detect_prot (ProtPosCol,CaseArg,x,y,z,i,xrefpt,yrefpt,zrefpt);
                                    break;
                                  default: break;
                                }
                              }
                              for (i=0; i<4; i++) {
                                 CaseArg = alka[x][y][z][i] + 2;
                                 switch (CaseArg) {
                                   case 1:
                detect_alka (AlkaNegCol,CaseArg,x,y,z,i,xrefpt,yrefpt,zrefpt);
                                    break;
                                  case 3:
                detect_alka (AlkaPosCol,CaseArg,x,y,z,i,xrefpt,yrefpt,zrefpt);
                                    break;
                                  default: break;
                                }
                              }
                           }
                        }
                     }
                  }

make_picture_current(Display);
}

/*****************************************************************************/
/*   Function Name:   detect_prot()                                          */
/*                                                                           */
/*   Input:   Color - An integer number of color                             */
/*            Case - An integer indicating the kind of particle              */
/*            Cx - The coordinate of x                                       */
/*            Cy - The coordinate of y                                       */
/*            Cz - The coordinate of z                                       */
/*            count - The number of the neighbors                            */
/*            xrefpt - The value of reference coordinate on x                */
/*            yrefpt - The value of reference coordinate on y                */
/*            zrefpt - The value of reference coordinate on z                */
/*                                                                           */
/*   Output:  The line links to its avaiable neighbors                       */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function detect the neighbors of a proton and links all its     */
/*   neighbors which have same kind of proton particles together.            */
/*************************************.**************:*********************/

void detect_prot(Color,Case,cx,cy,cz,count,refpt1,refpt2,refpt3)
int    Color, Case, count, cx, cy, cz;
float  refpt1, refpt2, refpt3;
{
   int j;

     move3d (Display, refpt1 + xshift[count],
                      refpt2 + yshift[count],
                      refpt3 + zshift[count]);
   for (j=count+1; j<8; j++) {
     if ((prot[cx][cy][cz][j]+2) == Case) {
        line_color_index (Display, Color);
        draw3d (Display, refpt1 + xshift[j],
                         refpt2 + yshift[j],
                         refpt3 + zshift[j]);
        move3d (Display, refpt1 + xshift[count],
                         refpt2 + yshift[count],
```

```
                        refpt3 + zshift[count]);
    }
  }
}

/*********************************************************************/
/*   Function Name:   detect_alka()                                  */
/*                                                                   */
/*   Input:   Color - An integer of color                            */
/*            Case - An integer indicating the kind of particle      */
/*            cx - The x coordinate                                  */
/*            cy - The y coordinate                                  */
/*            cz - The z coordinate                                  */
/*            count - The number of neighbors                        */
/*            refpt1 - The coordinate of reference on x              */
/*            refpt2 - The coordinate of reference on y              */
/*            refpt3 - The coordinate of reference on z              */
/*                                                                   */
/*   Output:  Linked lines for its proper neighbors                  */
/*                                                                   */
/*   Return:  None                                                   */
/*                                                                   */
/*   Description:                                                    */
/*      This function draws lines between its neighbors for alkala,  */
/*   whose neighbors are same kind of particles.                     */
/*********************************************************************/

void detect_alka(Color, Case, cx, cy, cz, count,refpt1,refpt2,refpt3)
int Case, Color, count, cx, cy, cz;
float refpt1, refpt2, refpt3;
{
   int  j;

     move3d (Display, refpt1 + xshift[count+8],
                      refpt2 + yshift[count+8],
                      refpt3 + zshift[count+8]);
   for (j=count+1; j<4; j++) {
     if ((alka[cx][cy][cz][j]+2) == Case) {
        line_color_index (Display, Color);
        draw3d (Display, refpt1 + xshift[j+8],
                         refpt2 + yshift[j+8],
                         refpt3 + zshift[j+8]);
        move3d (Display, refpt1 + xshift[count+8],
                         refpt2 + yshift[count+8],
                         refpt3 + zshift[count+8]);
    }
  }
}

/*********************************************************************/
/*   Function Name:   proton_line()                                  */
/*                                                                   */
/*   Input:   None                                                   */
/*                                                                   */
/*   Output:  The linked lines for proton                            */
/*                                                                   */
/*   Return:  None                                                   */
/*                                                                   */
/*   Description:                                                    */
/*      This function draws lines between same kind of proton and draws */
/*   marks for alkala.                                               */
/*********************************************************************/

void proton_line()
{
        int     x, y, z, i, Buffer=0;
```

```
            ClearScreen(Display);
            hidden_surface(Display, HSRoff, CULLoff);
            double_buffer(Display, TRUE, 8);

             for (Theta = 0.0; Theta < 360.1; Theta += 5.0) {
                Camera.camx = 5 * cos((Theta+70) deg);
                Camera.camz = 3.0;
                Camera.camy = 5   * sin((Theta+70) deg);

                   view_camera(Display, &Camera);

                   dbuffer_switch(Display, Buffer = !Buffer);

                   for (x = 0; x < 8; x++) {
                      xrefpt = (float)x / 8.0;
                      for (y =0; y < 8; y++) {
                         yrefpt = (float)y / 8.0;
                         for (z =0; z < 8; z++) {
                            zrefpt = (float)z /8.0;
                            for (i = 0; i < 8; i++) {
                                CaseArg = prot[x][y][z][i] + 2;
                                switch (CaseArg) {
                                   case 1:
                                      detect_prot (ProtNegCol,CaseArg,x,y,z,i,xrefpt,y
                                      break;
                                   case 3:
                                      detect_prot (ProtPosCol,CaseArg,x,y,z,i,xrefpt,y
                                      break;
                                   default: break;
                                }
                            }
                            for (i= 0; i < 4; i++) {
                                CaseArg = alka[x][y][z][i] + 2;
                                switch (CaseArg) {
                                   case 1:
                                      drawmark(Display, xrefpt + xshift[i+8],
                                                        yrefpt + yshift[i+8],
                                                        zrefpt + zshift[i+8],
                                                        AlkaNegCol);
                                      break;
                                   case 3:
                                      drawmark(Display, xrefpt + xshift[i+8],
                                                        yrefpt + yshift[i+8],
                                                        zrefpt + zshift[i+8],
                                                        AlkaPosCol);
                                      break;
                                   default: break;
                                }
                            }
                         }
                      }
                   }
             }

make_picture_current(Display);
}

/*************************************************************************/
/*   Function Name:   alkala_line()                                      */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  The linked lines for alkala                                */
/*                                                                       */
/*   Return:  None                                                       */
```

```
/*                                                                          */
/*   Description:                                                           */
/*      This function draws lines between all the alkala and draws          */
/*   marks for the proton.                                                  */
/****************************************************************************/

void alkala_line()
{
        int     x, y, z,i, Buffer=0;


        ClearScreen(Display);
        hidden_surface(Display, HSRoff, CULLoff);
        double_buffer(Display, TRUE, 8);

          for (Theta = 0.0; Theta < 360.1; Theta += 5.0) {
             Camera.camx = 5 * cos((Theta+70) deg);
             Camera.camz = 3.0;
             Camera.camy = 5   * sin((Theta+70) deg);

                view_camera(Display, &Camera);

                dbuffer_switch(Display, Buffer = !Buffer);

                for (x = 0; x < 8; x++) {
                   xrefpt = (float)x / 8.0;
                   for (y =0; y < 8; y++) {
                      yrefpt = (float)y / 8.0;
                      for (z =0; z < 8; z++) {
                         zrefpt = (float)z /8.0;
                         for (i = 0; i < 8; i++) {
                            CaseArg = prot[x][y][z][i] + 2;
                            switch (CaseArg) {
                               case 1:
                                  drawmark (Display, xrefpt + xshift[i],
                                                     yrefpt + yshift[i],
                                                     zrefpt + zshift[i],
                                                     ProtNegCol);
                                  break;
                               case 3:
                                  drawmark (Display, xrefpt + xshift[i],
                                                     yrefpt + yshift[i],
                                                     zrefpt + zshift[i],
                                                     ProtPosCol);
                                  break;
                               default: break;
                            }
                         }
                         for (i = 0; i < 4; i++) {
                            CaseArg = alka[x][y][z][i] + 2;
                            switch (CaseArg) {
                              case 1:
        detect_alka(AlkaNegCol,CaseArg,x,y,z,i,xrefpt,yrefpt,zrefpt);
                                 break;
                              case 3:
        detect_alka(AlkaPosCol,CaseArg,x,y,z,i,xrefpt,yrefpt,zrefpt);
                                 break;
                              default: break;
                            }
                         }
                      }
                   }
                }
          }
make_picture_current(Display);
}
```

```
/*************************************************************************/
/*   File Name:   drawsphere.c                  Author:   Fan Yang       */
/*                                                                       */
/*   Files used with this package:   drawsphere.c                        */
/*                                   construct.c                         */
/*                                   tools.c                             */
/*                                   drawline.c                          */
/*                                   showlocal.c                         */
/*                                   spheres.c                           */
/*                                   newdip.c                            */
/*                                                                       */
/*   Functions in this file:   whole_sphere()                            */
/*                             proton_sphere()                           */
/*                             drawmark()                                */
/*                             alkala_sphere()                           */
/*************************************************************************/
#include <starbase.c.h>
#include <stdio.h>
#include <math.h>

#include "whole.h"

int display, colorval;

/*************************************************************************/
/*   Function Name:   whole_sphere                                       */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Display spheres for both proton and alkala                 */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function draws spheres for both proton and alkala in        */
/*   different colors.                                                   */
/*************************************************************************/

void whole_sphere()
{
        float   centers[128][3];
        int     colors[128];
        int     x, y, z, i;

             ClearScreen(Display);
             Theta = 0.0;
             Camera.camx = 5 * cos((Theta+70) deg);
             Camera.camz = 3.0;
             Camera.camy = 5   * sin((Theta+70) deg);

                view_camera(Display, &Camera);

                Strips = hidden_surface(Display, HSRon, CULLon);
                for (Strip = 1; Strip <= Strips; Strip++) {
                        zbuffer_switch(Display, Strip);

                for (x = 0; x < 8; x++) {
                   xrefpt = (float)x / 8.0;
                   for (y =0; y < 8; y++) {
                      yrefpt = (float)y / 8.0;
                      for (z =0; z < 8; z++) {
                         zrefpt = (float)z /8.0;
                         for (i = 0; i < 8; i++) {
                            fill_color_index (Display, 200+prot[x][y][z][i]);
                            DrawSphere (Display, xrefpt + xshift[i],
                                                 yrefpt + yshift[i],
```

```
                                                zrefpt + zshift[i],
                                                radius1);
                                    }
                                    for (i = 0; i < 4; i++) {
                                       fill_color_index (Display, 150+alka[x][y][z][i]);
                                       DrawSphere (Display, xrefpt + xshift[i + 8],
                                                yrefpt + yshift[i + 8],
                                                zrefpt + zshift[i + 8],
                                                radius2);
                                    }
                                 }
                              }
                           }
                        }
make_picture_current(Display);
return;
}

/*************************************************************************/
/*   Function Name:   proton_sphere()                                    */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Display proton as spheres on the screen                    */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function draws spheres for proton and draws marks for alkala.  */
/*************************************************************************/

void proton_sphere()
{
   int x, y, z,i, Buffer=0;
   void drawmark();

   ClearScreen(Display);
   Theta = 0.0;
   Camera.camx = 5 * cos((Theta+70) deg);
   Camera.camz = 3.0;
   Camera.camy = 5   * sin((Theta+70) deg);
   view_camera(Display, &Camera);
   Strips = hidden_surface(Display, HSRon, CULLon);
   for (Strip = 1; Strip <= Strips; Strip++) {
        zbuffer_switch(Display, Strip);

   for (x = 0; x < 8; x++) {
       xrefpt = (float)x/8.0;
       for (y = 0; y < 8; y++) {
           yrefpt = (float)y/8.0;
           for (z = 0; z < 8; z++) {
               zrefpt = (float)z/8.0;
               for (i = 0; i < 8 ; i++) {
                         fill_color_index (Display,prot[x][y][z][i]+200);
                         DrawSphere (Display, xrefpt + xshift[i],
                                              yrefpt + yshift[i],
                                              zrefpt + zshift[i], radius3);
                }
                for (i = 0; i < 4; i++) {
                    drawmark (Display, xrefpt + xshift[i],
                                       yrefpt + yshift[i],
                                       zrefpt + zshift[i],
                                       alka[x][y][z][i]+150);
                }
           }
       }
```

```
    }
  }

make_picture_current(Display);
return;
}

/*****************************************************************************/
/*    Function Name:   drawmark()                                            */
/*                                                                           */
/*    Input:   display - A descriptor of file                                 */
/*             xpos - The coordinate of x                                    */
/*             ypos - The coordinate of y                                    */
/*             zpos - The coordinate of z                                    */
/*             colorval - An integer of color                                */
/*                                                                           */
/*    Output:  A mark on the screen                                          */
/*                                                                           */
/*    Return:  None                                                          */
/*                                                                           */
/*    Description:                                                           */
/*       This function draws mark on the screen.                             */
/*****************************************************************************/

void drawmark(display, xpos, ypos, zpos, coloval)
int display, coloval;
float xpos, ypos, zpos;
{
   line_color_index (display, coloval);
   move3d (display, xpos, ypos, zpos);
   draw3d (display, xpos, ypos, zpos);
   return;
}

/*****************************************************************************/
/*   Function Name:   alkala_sphere()                                        */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Display spheres for alkala on the screen                       */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function draws spheres for alkala  and draws marks for proton. */
/*****************************************************************************/

void alkala_sphere()
{
        float   centers[128][3];
        int     colors[128];
        int     x, y, z, Buffer=0, i;

        ClearScreen (Display);
             Theta = 0.0;
             Camera.camx = 5 * cos((Theta+70) deg);
             Camera.camz = 3.0;
             Camera.camy = 5   * sin((Theta+70) deg);

                view_camera(Display, &Camera);
                Strips = hidden_surface(Display, HSRon, CULLon);
                for (Strip = 1; Strip <= Strips; Strip++) {
                        zbuffer_switch(Display, Strip);

                for (x = 0; x < 8; x++) {
                   xrefpt = (float)x / 8.0;
```

```
                    for (y =0; y < 8; y++) {
                       yrefpt = (float)y / 8.0;
                       for (z =0; z < 8; z++) {
                          zrefpt = (float)z /8.0;
                          for (i = 0; i < 8; i++) {
                             drawmark    (Display, xrefpt + xshift[i],
                                                   yrefpt + yshift[i],
                                                   zrefpt + zshift[i],
                                                   prot[x][y][z][i]+200);
                          }
                          for (i = 0; i < 4; i++) {
                             fill_color_index (Display, 150+alka[x][y][z][i]);
                             DrawSphere (Display, xrefpt + xshift[i + 8],
                                                   yrefpt + yshift[i + 8],
                                                   zrefpt + zshift[i + 8],
                                                   radius2);
                          }
                       }
                    }
                 }
              }
make_picture_current(Display);
return;
}
```

```
/*****************************************************************************/
/*   File Name:   showlocal.c                    Author:   Fan Yang          */
/*                                                                           */
/*   Files used with this package:   showlocal.c                             */
/*                                   construct.c                             */
/*                                   tools.c                                 */
/*                                   drawsphere.c                            */
/*                                   drawline.c                              */
/*                                   newdip.c                                */
/*                                   spheres.c                               */
/*                                                                           */
/*   Functions in this file:   showlocal()                                   */
/*                             TakeChoice()                                  */
/*                             TakeXMenu()                                   */
/*                             TakeYMenu()                                   */
/*                             TakeZMenu()                                   */
/*                             draw_cubic()                                  */
/*                             DrawSmCubic()                                 */
/*                             local_line()                                  */
/*                             RealLink()                                    */
/*                             DrawXaxis()                                   */
/*                             DrawYaxis()                                   */
/*                             DrawZaxis()                                   */
/*                             DrawXYaxis()                                  */
/*                             DrawXZaxis()                                  */
/*                             DrawYZaxis()                                  */
/*                             DrawXYZaxis()                                 */
/*                             XSphere()                                     */
/*                             YSphere()                                     */
/*                             ZSphere()                                     */
/*                             XYSphere()                                    */
/*                             XZSphere()                                    */
/*                             YZSphere()                                    */
/*                             XYZSphere()                                   */
/*                             RealSphere()                                  */
/*****************************************************************************/

#include <starbase.c.h>
#include <stdio.h>
#include <math.h>

#include "whole.h"

void drawline();
void drawsphere();
void ResetView ();
int  xt, yt, zt;

/*****************************************************************************/
/*   Function Name:   showlocal()                                            */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Display menus on the screen                                    */
/*                                                                           */
/*   Result:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function displays three menus one by one on the screen,         */
/*   and provides choices for x coordinate, y coordinete and z coordinate    */
/*   respectively.                                                           */
/*****************************************************************************/

void showlocal()
{
   void    PutMenu(), ShowChoice();
```

```
void    local_line(), local_sphere();
void    RealSphere();
Boolean TakeChoice(), TakeXMenu(), TakeYMenu(), TakeZMenu();
MenuSt  LoadMenu(), *GetChoice();
int     Choice,OldChoice = NO_CHOICE;
MenuSt  XMenu, YMenu, ZMenu, PMenu;
char    MenuFileName[80];
Boolean MenuDone, XMenuDone, YMenuDone, ZMenuDone;


  strcpy(MenuFileName, "XMenu.menu");
  GET_PATH (MENU_PATH, MenuFileName);
  XMenu = LoadMenu (MenuFileName);
  DelAllMesg (Overlay);
  PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - XMenu.Width, 0, &XMenu);
  do {
     while (GetChoice (Mouse, Overlay, &Choice) != &XMenu);
     ShowChoice (Overlay, Choice,OldChoice,  &XMenu);
     DelAllMesg (Overlay);
     OldChoice = Choice;
     XMenuDone = TakeXMenu (Choice);
     if (XMenuDone) {
         PutMenu (Overlay, NULL_STRING, RIGHT_EDGE-XMenu.Width, 0, &XMenu);
         ShowChoice (Overlay, Choice, NO_CHOICE, &XMenu);
     }
  } while (Choice != XMenu.NbrEnts);
  ClearScreen (Overlay);

  strcpy (MenuFileName, "YMenu.menu");
  GET_PATH (MENU_PATH, MenuFileName);
  YMenu = LoadMenu (MenuFileName);
  DelAllMesg (Overlay);
  OldChoice = NO_CHOICE;
  PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - YMenu.Width, 0, &YMenu);
  do {
     while (GetChoice (Mouse, Overlay, &Choice) != &YMenu);
     ShowChoice (Overlay, Choice, OldChoice,  &YMenu);
     DelAllMesg (Overlay);
     OldChoice = Choice;
     YMenuDone = TakeYMenu (Choice);
     if (YMenuDone) {
         PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - YMenu.Width, 0, &YMenu);
         ShowChoice (Overlay, Choice, NO_CHOICE, &YMenu);
     }
  } while (Choice != YMenu.NbrEnts);
  ClearScreen (Overlay);

  strcpy (MenuFileName, "ZMenu.menu");
  GET_PATH (MENU_PATH, MenuFileName);
  ZMenu = LoadMenu (MenuFileName);
  DelAllMesg (Overlay);
  OldChoice = NO_CHOICE;
  PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - ZMenu.Width, 0, &ZMenu);
  do {
     while (GetChoice (Mouse, Overlay, &Choice) != &ZMenu);
     ShowChoice (Overlay, Choice, OldChoice,  &ZMenu);
     DelAllMesg (Overlay);
     OldChoice = Choice;
     ZMenuDone = TakeZMenu (Choice);
     if (ZMenuDone) {
         PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - ZMenu.Width, 0, &ZMenu);
         ShowChoice (Overlay, Choice, NO_CHOICE, &ZMenu);
     }
  } while (Choice != ZMenu.NbrEnts);

ClearScreen (Overlay);
```

```
      strcpy (MenuFileName, "PMenu.menu");
      GET_PATH (MENU_PATH, MenuFileName);
      PMenu = LoadMenu (MenuFileName);
      DelAllMesg (Overlay);
      OldChoice = NO_CHOICE;
      PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - PMenu.Width, 0, &PMenu);
      do
      {
         while (GetChoice (Mouse, Overlay, &Choice) != &PMenu);
         ShowChoice (Overlay, Choice,OldChoice, &PMenu);
         DelAllMesg (Overlay);
         OldChoice = Choice;
         MenuDone = TakeChoice (Choice);
         if (MenuDone) {
            PutMenu (Overlay, NULL_STRING, RIGHT_EDGE-PMenu.Width,0, &PMenu);
            ShowChoice (Overlay, Choice, NO_CHOICE, &PMenu);
         }
       } while (Choice != PMenu.NbrEnts);
/* ResetView (Display); */
ClearScreen (Overlay);
/* ClearScreen (Display); */
RestoreActiveMenus (Overlay);
}

/*************************************************************************/
/*   Function Name:   TakeChoice()                                       */
/*                                                                       */
/*   Input:   Choice - A pointer to menu choice                          */
/*                                                                       */
/*   Output:  None                                                       */
/*                                                                       */
/*   Return:  Gone - A Boolean variable                                  */
/*                                                                       */
/*   Description:                                                        */
/*      This function provides the choices for the showlocal.c, it       */
/*   includes using line and sphere models for local structures.         */
/*************************************************************************/

Boolean TakeChoice (Choice)
{
   void local_line(), local_sphere();
   void DrawXaxis(), DrawYaxis(), DrawZaxis();
   void DrawXYaxis(), DrawXZaxis(), DrawYZaxis(),DrawXYZaxis();
   void XSphere(), YSphere(), ZSphere(), XYSphere();
   void XZSphere(), YZSphere(), XYZSphere();
   Boolean Gone = FALSE;

   switch (Choice) {
      case 1: Gone = TRUE;
              if ((xt<8) & (yt<8) & (zt<8))
                 local_line();
              else if ((xt>8) & (yt<8) & (zt<8))
                     DrawXaxis();
                     else if ((xt>8) & (yt>8) & (zt<8))
                        DrawXYaxis();
                        else if ((xt>8) & (yt>8) & (zt>8))
                           DrawXYZaxis();
                           else if ((xt<8) & (yt>8) & (zt<8))
                              DrawYaxis();
                              else if ((xt<8) & (yt>8) & (zt>8))
                                 DrawYZaxis();
                                 else if ((xt<8) & (yt<8) & (zt>8))
                                    DrawZaxis();
                                    else if ((xt>8) & (yt<8) & (zt>8))
                                       DrawXZaxis();
```

```
                break;
        case 2: Gone = TRUE;
                if ((xt<8) & (yt<8) & (zt<8))
                    local_sphere();
                    else if ((xt>8) & (yt<8) & (zt<8))
                         XSphere();
                         else if ((xt>8) & (yt>8) & (zt<8))
                              XYSphere();
                              else if ((xt>8) & (yt>8) & (zt>8))
                                   XYZSphere();
                                   else if ((xt<8) & (yt>8) & (zt<8))
                                        YSphere();
                                        else if ((xt<8) & (yt>8) & (zt>8))
                                             YZSphere();
                                             else if ((xt>8) & (yt<8) & (zt>8))
                                                  XZSphere();
                                                  else if ((xt<8) & (yt<8) & (zt>8))
                                                       ZSphere();
                  break;
        case 3: break;
    }
    return (Gone);
}

/*****************************************************************************/
/*    Function Name:    TakeXMenu ()                                         */
/*                                                                           */
/*    Input:    Choice - A pointer to menu choices                         */
/*                                                                           */
/*    Output:   None                                                         */
/*                                                                           */
/*    Return:   Gone - A Boolean variable                                    */
/*                                                                           */
/*    Description:                                                           */
/*       This function provides the choices for the menu to x coordinate.    */
/*****************************************************************************/

Boolean TakeXMenu (Choice)
{
    Boolean Gone = FALSE;

    switch (Choice ) {
       case 1: Gone = TRUE; xt=0; break;
       case 2: Gone = TRUE; xt=1; break;
       case 3: Gone = TRUE; xt=2; break;
       case 4: Gone = TRUE; xt=3; break;
       case 5: Gone = TRUE; xt=4; break;
       case 6: Gone = TRUE; xt=5; break;
       case 7: Gone = TRUE; xt=6; break;
       case 8: Gone = TRUE; xt=7; break;
       case 9: Gone = TRUE; xt=999; break;
       case 10: break;
    }
    return (Gone);
}

/*****************************************************************************/
/*    Function Name:    TakeYMenu ()                                         */
/*                                                                           */
/*    Input:    Choice - A pointer to menu choice                          */
/*                                                                           */
/*    Output:   None                                                         */
/*                                                                           */
/*    Return:   Gone - A Boolean variable                                    */
/*                                                                           */
/*    Description:                                                           */
```

```
/*      This function provides the choices for the menu to y coordinate.     */
/*****************************************************************************/

Boolean TakeYMenu (Choice)
{
    Boolean Gone = FALSE;

    switch (Choice) {
       case 1: Gone = TRUE; yt=0; break;
       case 2: Gone = TRUE; yt=1; break;
       case 3: Gone = TRUE; yt=2; break;
       case 4: Gone = TRUE; yt=3; break;
       case 5: Gone = TRUE; yt=4; break;
       case 6: Gone = TRUE; yt=5; break;
       case 7: Gone = TRUE; yt=6; break;
       case 8: Gone = TRUE; yt=7; break;
       case 9: Gone = TRUE; yt=999; break;
       case 10: break;
    }
    return (Gone);
}

/*****************************************************************************/
/*   Function Name:   TakeZMenu ()                                           */
/*                                                                           */
/*   Input:   Choice - A pointer to menu choices                             */
/*                                                                           */
/*   Output:  None                                                           */
/*                                                                           */
/*   Return:  Gone - A Boolean variable                                      */
/*                                                                           */
/*   Description:                                                            */
/*      This function provides the choices for the menu to z coordinate.     */
/*****************************************************************************/

Boolean TakeZMenu (Choice)
{
    Boolean Gone = FALSE;

    switch (Choice) {
       case 1: Gone = TRUE; zt=0; break;
       case 2: Gone = TRUE; zt=1; break;
       case 3: Gone = TRUE; zt=2; break;
       case 4: Gone = TRUE; zt=3; break;
       case 5: Gone = TRUE; zt=4; break;
       case 6: Gone = TRUE; zt=5; break;
       case 7: Gone = TRUE; zt=6; break;
       case 8: Gone = TRUE; zt=7; break;
       case 9: Gone = TRUE; zt=999; break;
       case 10: break;
    }
    return (Gone);
}

/*****************************************************************************/
/*   Function Name:   draw_cubic()                                           */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Display a cube on the screen                                   */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function draws a big cube as a border of the model.             */
/*****************************************************************************/
```

```
void draw_cubic()
{
   line_color_index (Display,155);
   move3d (Display, 0.0, 0.0, 0.0);
   draw3d (Display, 0.0, 1.0, 0.0);
   draw3d (Display, 1.0, 1.0, 0.0);
   draw3d (Display, 1.0, 0.0, 0.0);
   draw3d (Display, 0.0, 0.0, 0.0);
   draw3d (Display, 0.0, 0.0, 1.0);
   draw3d (Display, 0.0, 1.0, 1.0);
   draw3d (Display, 1.0, 1.0, 1.0);
   draw3d (Display, 1.0, 0.0, 1.0);
   draw3d (Display, 0.0, 0.0, 1.0);
   move3d (Display, 0.0, 1.0, 0.0);
   draw3d (Display, 0.0, 1.0, 1.0);
   move3d (Display, 1.0, 1.0, 0.0);
   draw3d (Display, 1.0, 1.0, 1.0);
   move3d (Display, 1.0, 0.0, 0.0);
   draw3d (Display, 1.0, 0.0, 1.0);
}

/*****************************************************************************/
/*   Function Name:   DrawSmCubic()                                          */
/*                                                                           */
/*   Input:   xval1 - The relative coordinate of the first point on x        */
/*            yval1 - The relative coordinate of the first point on y        */
/*            zval1 - The relative coordinate of the first point on z        */
/*            xval2 - The relative coordinate of the second point on x       */
/*            yval2 - The relative coordinate of the second point on y       */
/*            zval2 - The relative coordiante of the second point on z       */
/*                                                                           */
/*   Output:  The small reference cube on the screen                         */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function draws a small reference cube as a cell for 12 voxel    */
/*   group.                                                                  */
/*****************************************************************************/

void DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2)
float xval1,yval1,zval1,xval2,yval2,zval2;
{
   line_color_index (Display, 79);
   move3d (Display, xval1, yval1, zval1);
   draw3d (Display, xval1, yval2, zval1);
   draw3d (Display, xval2, yval2, zval1);
   draw3d (Display, xval2, yval1, zval1);
   draw3d (Display, xval1, yval1, zval1);
   draw3d (Display, xval1, yval1, zval2);
   draw3d (Display, xval1, yval2, zval2);
   draw3d (Display, xval2, yval2, zval2);
   draw3d (Display, xval2, yval1, zval2);
   draw3d (Display, xval1, yval1, zval2);
   move3d (Display, xval1, yval2, zval1);
   draw3d (Display, xval1, yval2, zval2);
   move3d (Display, xval2, yval2, zval1);
   draw3d (Display, xval2, yval2, zval2);
   move3d (Display, xval2, yval1, zval1);
   draw3d (Display, xval2, yval1, zval2);
}

/*****************************************************************************/
/*   Function Name:   local_line()                                           */
/*                                                                           */
```

```
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  Display a cell cube and 12 voxels linked with lines           */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function calls the DrawSmCubic to draw a small reference cube  */
/*   for a cell and shows the relationship of 12 voxels with linked lines.  */
/**************************************************************************/

void local_line()
{
   void draw_cubic();
   void DrawSmCubic();
   void RealLink ();
   int Buffer=0;
   float xval1,yval1,zval1,xval2,yval2,zval2;

      xrefpt = (float)xt /8.0;
      yrefpt = (float)yt /8.0;
      zrefpt = (float)zt /8.0;
      ClearScreen(Display);
      hidden_surface (Display, HSRoff, CULLoff);
      double_buffer (Display,TRUE, 8);

      for (Theta=0.0; Theta<360.1; Theta+=2.0) {
          Camera.camx = 3*cos((Theta+50) deg);
          Camera.camz = 4.0;
          Camera.camy = 3*sin((Theta+50) deg);
          view_camera (Display, &Camera);
          dbuffer_switch (Display, Buffer = !Buffer);
      draw_cubic ();
      xval1 = xt*0.125;
      yval1 = yt*0.125;
      zval1 = zt*0.125;
      xval2 = (1+xt)*0.125;
      yval2 = (1+yt)*0.125;
      zval2 = (1+zt)*0.125;
      DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);
      RealLink (xt,yt,zt);
      }
      make_picture_current(Display);
}

/**************************************************************************/
/*   Function Name:   RealLink()                                          */
/*                                                                        */
/*   Input:   xt - The x coordinate of parameter                          */
/*            yt - The y coordinate of parameter                          */
/*            zt - The z coordinate of parameter                          */
/*                                                                        */
/*   Output:  None                                                        */
/*                                                                        */
/*   Return:  None                                                        */
/*                                                                        */
/*   Description:                                                         */
/*      This function calls detect_prot and detect_alka to detect the same */
/*   kind of particle in its neighbors in the same cell.                  */
/**************************************************************************/

void RealLink(xt,yt,zt)
int xt,yt,zt;
{
   int i;
      for (i=0; i < 8; i++) {
```

```
            CaseArg =  prot[xt][yt][zt][i] + 2;
            switch (CaseArg) {
               case 1: detect_prot (ProtNegCol,CaseArg,xt,yt,zt,i,xrefpt,yrefpt,zrefpt
                       break;
               case 3: detect_prot (ProtPosCol,CaseArg,xt,yt,zt,i,xrefpt,yrefpt,zrefpt
                       break;
               default: break;
            }
         }
         for (i=0; i < 4; i++) {
            CaseArg = alka[xt][yt][zt][i] + 2;
            switch (CaseArg) {
               case 1: detect_alka (AlkaNegCol,CaseArg,xt,yt,zt,i,xrefpt,yrefpt,zrefpt
                       break;
               case 3: detect_alka (AlkaPosCol,CaseArg,xt,yt,zt,i,xrefpt,yrefpt,zrefpt
                       break;
               default: break;
            }
         }
}

/*************************************************************************/
/*   Function Name:   DrawXaxis()                                        */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Slice cube and 8 cells with 8 * 12 voxels linked with lines */
/*            along x coordinate                                         */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function calls DrawSmCubic to draw a reference slice cube and */
/*   draws the relationship of 8 * 12 voxels in 8 cells with linked lines */
/*   along the x coordinate.                                             *//**

void DrawXaxis ()
{
    void draw_cubic();
    void RealLink();
    void DrawSmCubic();
    int x, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen (Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE,8);
    for (Theta = 0.0; Theta < 360.1; Theta += 2.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 4.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubic ();
    xval1 = 0.0;
    yval1 = yt*0.125;
    zval1 = zt*0.125;
    xval2 = 1.0;
    yval2 = (1+yt)*0.125;
    zval2 = (1+zt)*0.125;
    DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);

    yrefpt = (float)yt /8.0;
    zrefpt = (float)zt /8.0;
    for (x=0; x<8; x++) {
       xrefpt = (float)x /8.0;
```

```
        RealLink (x, yt, zt);
      }
     }
     make_picture_current (Display);
}

/*************************************************************************/
/*   Function Name:   DrawYaxis()                                        */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Slice cube and the relationship of 8 * 12 voxels in 8 cells */
/*            with linked line along y coordinate.                       */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function calls DrawSmCubic to draw a slice reference cube and */
/*   draws relationship of 8 * 12 voxels in 8 cells with linked lines along */
/*   y coordinate.                                                       */
/*************************************************************************/

void DrawYaxis ()
{
    void draw_cubic();
    void RealLink();
    void DrawSmCubic();

    int y, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen(Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE, 8);
    for (Theta =0.0; Theta < 360.1; Theta += 2.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 4.0;
        Camera.camy = 3*sin((Theta+50) deg);

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);

    draw_cubic ();
    xval1 = xt*0.125;
    yval1 = 0.0;
    zval1 = zt*0.125;
    xval2 = (1+xt)*0.125;
    yval2 = 1.0;
    zval2 = (1+zt)*0.125;
    DrawSmCubic (xval1, yval1, zval1, xval2, yval2, zval2);

    xrefpt = (float)xt /8.0;
    zrefpt = (float)zt /8.0;
    for (y=0; y<8; y++) {
       yrefpt = (float)y /8.0;
       RealLink (xt, y, zt);
    }
   }
    make_picture_current (Display);
}

/*************************************************************************/
/*   Function Name:   DrawZaxis()                                        */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
```

```
/*   Output:  Slice cube and the relationship of 8 * 12 voxels in 8 cells    */
/*            with linked lines in z coordinate                              */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubic to draw a slice reference cube along */
/*   z coordinate and draws the relationship of 8 * 12 voxels in 8 cells     */
/*   with linked lines.                                                      */
/*****************************************************************************/

void DrawZaxis ()
{
    void draw_cubic();
    void RealLink();
    void DrawSmCubic();
    int z, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen(Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE, 8);
    for (Theta = 0.0; Theta < 360.1; Theta += 2.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 4.0;
        Camera.camy = 3*sin((Theta+50) deg);

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubic();
    xval1 = xt*0.125;
    yval1 = yt*0.125;
    zval1 = 0.0;
    xval2 = (1+xt)*0.125;
    yval2 = (1+yt)*0.125;
    zval2 = 1.0;
    DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);

    xrefpt = (float)xt /8.0;
    yrefpt = (float)yt /8.0;
    for (z=0; z<8; z++) {
       zrefpt = (float)z /8.0;
       RealLink (xt, yt, z);
    }
   }
    make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   DrawXYaxis()                                           */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and relationship of 8 * 8 * 12 voxels in 8 * 8     */
/*            cells with linked lines                                        */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubic to draw a slice reference plane      */
/*   along x-y axes and draws the relationship of 8 * 8 * 12 voxels in       */
/*   8 * 8 cells with linked lines.                                          */
/*****************************************************************************/

void DrawXYaxis ()
{
```

```
    void draw_cubic();
    void RealLink();
    void DrawSmCubic();
    int x,y, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen (Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE, 8);
    for (Theta=0.0; Theta<360.1; Theta+=2.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 4.0;
        Camera.camy = 3*sin((Theta+50) deg);

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubic();
    xval1 = 0.0;
    yval1 = 0.0;
    zval1 = zt*0.125;
    xval2 = 1.0;
    yval2 = 1.0;
    zval2 = (1+zt)*0.125;
    DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);

    zrefpt = (float)zt /8.0;
    for (x=0; x<8; x++) {
       xrefpt = (float)x /8.0;
       for (y=0; y<8; y++) {
          yrefpt = (float)y /8.0;
          RealLink (x, y, zt);
       }
    }
   }
    make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   DrawXZaxis()                                           */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and relationship of 8 * 8 * 12 voxels in 8 * 8     */
/*            cells with linked lines                                        */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubic to draw a slice reference plane      */
/*   along x-z axes and draws the relationship of 8 * 8 * 12 voxels in       */
/*   8 * 8 cells with linked lines.                                          */
/*****************************************************************************/

void DrawXZaxis ()
{
    void draw_cubic();
    void RealLink();
    void DrawSmCubic();
    int x, z,Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen(Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE, 8);
    for (Theta=0.0; Theta<360.1; Theta+=2.0) {
        Camera.camx = 3*cos((Theta+50) deg);
```

```
        Camera.camz = 4.0;
        Camera.camy = 3*sin((Theta+50) deg);

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubic();
    xval1 = 0.0;
    yval1 = yt*0.125;
    zval1 = 0.0;
    xval2 = 1.0;
    yval2 = (1+yt)*0.125;
    zval2 = 1.0;
    DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);

    yrefpt = (float)yt /8.0;
    for (x=0; x<8; x++) {
       xrefpt = (float)x /8.0;
       for (z=0; z<8; z++) {
          zrefpt = (float)z /8.0;
          RealLink (x, yt, z);
       }
    }
   }
    make_picture_current(Display);
}

/*****************************************************************************/
/*   Function Name:   DrawYZaxis ()                                          */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and relationship of 8 * 8 * 12 voxels in 8 * 8     */
/*            cells in y-z axes                                              */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubic to draw a slice reference plane      */
/*   along y-z axes and draws the relationship of 8 * 8 * 12 voxels in       */
/*   8 * 8 cells with linked lines.                                          */
/*****************************************************************************/

void DrawYZaxis ()
{
    void draw_cubic();
    void RealLink();
    void DrawSmCubic();
    int y, z, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen (Display);
    hidden_surface(Display, HSRoff, CULLoff);
    double_buffer(Display, TRUE, 8);
    for (Theta=0.0; Theta<360.1; Theta+=2.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz= 4.0;
        Camera.camy = 3*sin((Theta+50) deg);

        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubic();
    xval1 = xt*0.125;
    yval1 = 0.0;
    zval1 = 0.0;
    xval2 = (1+xt)*0.125;
    yval2 = 1.0;
```

```
        zval2 = 1.0;
        DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);

        xrefpt = (float)xt/8.0;
        for (y=0; y<8; y++) {
            yrefpt = (float)y /8.0;
            for (z=0; z<8; z++) {
                zrefpt = (float)z /8.0;
            RealLink (xt, y, z);
            }
        }
    }
        make_picture_current (Display);
}

/*****************************************************************************/
/*    Function Name:    DrawXYZaxis()                                        */
/*                                                                           */
/*    Input:    None                                                         */
/*                                                                           */
/*    Output:   None                                                         */
/*                                                                           */
/*    Return:   None                                                         */
/*                                                                           */
/*    Description:                                                           */
/*       This function does nothing and return to the main menu for symmetry.*/
/*****************************************************************************/

void DrawXYZaxis ()
{
return;
}

/*****************************************************************************/
/*    Function Name:    local_sphere()                                       */
/*                                                                           */
/*    Input:    None                                                         */
/*                                                                           */
/*    Output:   Reference cube and spheres for voxels in a cell              */
/*                                                                           */
/*    Return:   None                                                         */
/*                                                                           */
/*    Description:                                                           */
/*       This function calls DrawSmCubic to draw a reference cube and draws */
/*    spheres for voxels in a cell.                                          */
/*****************************************************************************/

void local_sphere()
{
   static Boolean  Col;
   void draw_cubic ();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   void DrawSmCubic();
   int Buffer = 0;

   ClearScreen (Display);

   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=5.0) {
            Camera.camx = 3*cos((Theta + 50) deg);
            Camera.camz = 5.0;
            Camera.camy = 3*sin((Theta + 50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
        draw_cubic();
        xval1 = xt*0.125;
```

```
            yval1 = yt*0.125;
            zval1 = zt*0.125;
            xval2 = (1+xt)*0.125;
            yval2 = (1+yt)*0.125;
            zval2 = (1+zt)*0.125;
            DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);
            xrefpt = (float)xt /8.0;
            yrefpt = (float)yt /8.0;
            zrefpt = (float)zt /8.0;
            RealSphere (xt,yt,zt);
            hidden_surface (Display, HSRoff, CULLoff);
        }
        make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   XSphere()                                              */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice cube and spheres for voxels along x coordinate           */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubic to draw reference slice cube along   */
/*   x coordinate and draws 8 * 12 spheres for voxels in 8 cells.            */
/*****************************************************************************/

void XSphere()
{
   void draw_cubic ();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   void DrawSmCubic();
   int x, Buffer = 0;

   ClearScreen (Display);

   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=15.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
        draw_cubic();
        xval1 = 0.0;
        yval1 = yt*0.125;
        zval1 = zt*0.125;
        xval2 = 1.0;
        yval2 = (1+yt)*0.125;
        zval2 = (1+zt)*0.125;
        DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);
        yrefpt = (float)yt/8.0;
        zrefpt = (float)zt/8.0;
        for (x=0; x<8; x++) {
            xrefpt = (float)x/8.0;
            RealSphere(x,yt,zt);
    }
    hidden_surface (Display, HSRoff, CULLoff);
  }
    make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   YSphere()                                              */
```

```
/*                                                                          */
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  Slice cube and spheres for voxels along y coordinate          */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function calls DrawSmCubic to draw a slice cube along y        */
/*   coordinate and draws 8 * 12 spheres for voxels in 8 cells.             */
/****************************************************************************/

void YSphere()
{
   void draw_cubic ();
   void DrawSmCubic();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int y, Buffer = 0;

   ClearScreen (Display);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=15.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer =! Buffer);
        draw_cubic();
        xval1 = xt*0.125;
        yval1 = 0.0;
        zval1 = zt*0.125;
        xval2= (1+xt)*0.125;
        yval2 = 1.0;
        zval2 = (1+zt)*0.125;
        DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);
        xrefpt = (float)xt/8.0;
        zrefpt = (float)zt/8.0;
        for (y=0; y<8; y++) {
            yrefpt = (float)y/8.0;
            RealSphere(xt,y,zt);
       }
       hidden_surface (Display, HSRoff, CULLoff);
   }
       make_picture_current (Display);
}

/****************************************************************************/
/*   Function Name:   ZSphere()                                             */
/*                                                                          */
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  Slice cube and spheres along z coordinate                     */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function calls DrawSmCubic to draw a slice reference cube and  */
/*   draws 8 * 12 spheres in 8 cells.                                       */
/****************************************************************************/

void ZSphere ()
{
     void draw_cubic();
     void DrwmSmCubic();
     float xval1,yval1,zval1,xval2,yval2,zval2;
     int z, Buffer = 0;
```

```
      ClearScreen (Display);
      double_buffer (Display, TRUE, 8);
      for (Theta=0.0; Theta<360.1; Theta+=15.0) {
           Camera.camx = 3*cos((Theta+50) deg);
           Camera.camz = 5.0;
           Camera.camy = 3*sin((Theta+50) deg);
           view_camera (Display, &Camera);
           dbuffer_switch (Display, Buffer = !Buffer);
           draw_cubic();
           xval1 = xt*0.125;
           yval1 = yt*0.125;
           zval1 = 0.0;
           xval2 = (1+xt)*0.125;
           yval2 = (1+yt)*0.125;
           zval2 = 1.0;
           DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);
           xrefpt = (float)xt/8.0;
           yrefpt = (float)yt/8.0;
           for (z=0.0; z<8; z++) {
               zrefpt = (float)z/8.0;
               RealSphere (xt,yt,z);
           }
           hidden_surface (Display, HSRoff, CULLoff);
        }
           make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   XYSphere ()                                            */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 spheres along x-y coordinate        */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubic to draw a reference slice plane and  */
/*   draws 8 * 8 * 12 spheres in 8 * 8 cells along x-y coordinate.           */
/*****************************************************************************/

void XYSphere ()
{
   void draw_cubic();
   void DrawSmCubic ();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int x,y, Buffer = 0;

   ClearScreen (Display);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=30.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
        draw_cubic();
        xval1 =0.0;
        yval1 = 0.0;
        zval1 = zt*0.125;
        xval2 = 1.0;
        yval2 = 1.0;
        zval2 = (1+zt)*0.125;
        DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);
        zrefpt = (float)zt/8.0;
```

```
        for (x=0; x<8; x++) {
            xrefpt = (float)x/8.0;
            for (y=0; y<8; y++) {
                yrefpt = (float)y/8.0;
                RealSphere (x,y,zt);
            }
         }
         hidden_surface (Display, HSRoff, CULLoff);
    }
         make_picture_current(Display);
}

/*****************************************************************************/
/*   Function Name:   XZSphere ()                                            */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 spheres along x-z coordinate        */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicf to draw a reference slice plane and */
/*   draws 8 * 8 * 12 spheres in 8 * 8 cells along x-z coordinate.           */
/*****************************************************************************/

void XZSphere()
{
   void draw_cubic();
   void DrawSmCubic();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int x,z, Buffer = 0;

   ClearScreen (Display);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=30.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
        draw_cubic();
        xval1 = 0.0;
        yval1 = yt*0.125;
        zval1 = 0.0;
        xval2 = 1.0;
        yval2 = (1+yt)*0.125;
        zval2 = 1.0;
        DrawSmCubic (xval1,yval1,zval1,xval2,yval2,zval2);
        yrefpt = (float)yt/8.0;
        for (x=0; x<8; x++) {
            xrefpt = (float)x/8.0;
            for (z=0; z<8; z++) {
                 zrefpt = (float)z/8.0;
                 RealSphere(x,yt,z);
            }
         }
         hidden_surface (Display, HSRoff, CULLoff);
    }
    make_picture_current(Display);
}

/*****************************************************************************/
/*   Function Name:   YZSphere ()                                            */
/*                                                                           */
/*   Input:   None                                                           */
```

```
/*                                                                          */
/*   Output:  Slice plane and 8 * 8 * 12 spheres along y-z coordinate       */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function calls DrawSmCubic to draw a reference slice plane and */
/*   draws 8 * 8 * 12 spheres in 8 * 8 cells along y-z coordinate.          */
/****************************************************************************/

void YZSphere()
{
   void draw_cubic();
   void DrawSmCubic();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int y,z, Buffer = 0;

   ClearScreen(Display);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=30.0) {
       Camera.camx = 3*cos((Theta+50) deg);
       Camera.camz = 5.0;
       Camera.camy = 3*sin((Theta+50) deg);
       view_camera (Display, &Camera);
       dbuffer_switch (Display, Buffer = !Buffer);
       draw_cubic();
       xval1 = xt*0.125;
       yval1 = 0.0;
       zval1 = 0.0;
       xval2 = (1+xt)*0.125;
       yval2 = 1.0;
       zval2 = 1.0;
       xrefpt = (float)xt/8.0;
       for (y=0; y<8; y++) {
           yrefpt = (float)y/8.0;
           for (z=0; z<8; z++) {
               zrefpt = (float)z/8.0;
               RealSphere(xt,y,z);
           }
        }
       hidden_surface (Display, HSRoff, CULLoff);
     }
     make_picture_current (Display);
}

/****************************************************************************/
/*   Function Name:   XYZSphere ()                                          */
/*                                                                          */
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  None                                                          */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function does nothing but return to the main menu.             */
/****************************************************************************/

void XYZSphere ()
{
   return;
}

/****************************************************************************/
/*   Function Name:   RealSphere ()                                         */
/*
```

```
/*   Input:   xt - the x coordinate of parameter                                */
/*            yt - the y coordinate of parameter                                */
/*            zt - the z coordinate of parameter                                */
/*                                                                              */
/*   Output:  Sphere                                                            */
/*                                                                              */
/*   Return:  None                                                              */
/*                                                                              */
/*   Description:                                                               */
/*      This function draws sphere according to the parameters of coordinate.  */
/********************************************************************************/

void RealSphere(xt,yt,zt)
int xt,yt,zt;
{
   int Col;
   int i;

         Strips = hidden_surface (Display, HSRon, CULLon);
         for (Strip=1; Strip<=Strips; Strip++) {
            zbuffer_switch (Display, Strip);
            for (i=0; i<8; i++) {
               Col = prot[xt][yt][zt][i] + 200;
               fill_color_index (Display, Col);
               DrawSphere (Display, xrefpt + xshift[i],
                                    yrefpt + yshift[i],
                                    zrefpt + zshift[i],
                                    radius1);
             }
             for (i=0; i<4; i++) {
                Col = alka[xt][yt][zt][i]+150;
                fill_color_index (Display, Col);
                DrawSphere (Display, xrefpt + xshift[i+8],
                                     yrefpt + yshift[i+8],
                                     zrefpt + zshift[i+8],
                                     radius2);
             }
          }
}
```

```
/*************************************************************************/
/*   File Name:   newdip.c                    Author:   Fan Yang         */
/*                                                                       */
/*   File used with this package :   newdip.c                            */
/*                                   showlocal.c                         */
/*                                   construct.c                         */
/*                                   drawspheres.c                       */
/*                                   drawline.c                          */
/*                                   drawsphere.c                        */
/*                                   tools.c                             */
/*                                                                       */
/*   Functions in this file:   newdip ()                                 */
/*                             TakeChoiceNew ()                          */
/*                             TakeXMenuNew ()                           */
/*                             TakeYMenuNew ()                           */
/*                             TakeZMenuNew ()                           */
/*                             draw_cubicNew ()                          */
/*                             DrawSmCubicNew ()                         */
/*                             new_cube ()                               */
/*                             RealLinkNew ()                            */
/*                             XaxisCube ()                              */
/*                             YaxisCube ()                              */
/*                             ZaxisCube ()                              */
/*                             XYaxisCube ()                             */
/*                             XZaxisCube ()                             */
/*                             YZaxisCube ()                             */
/*                             XYZaxisCube ()                            */
/*                             new_pyramid ()                            */
/*                             XPyramid ()                               */
/*                             YPyramid ()                               */
/*                             ZPyramid ()                               */
/*                             XYPyramid ()                              */
/*                             XZPyramid ()                              */
/*                             YZPyramid ()                              */
/*                             XYZPyramid ()                             */
/*                             RealPyramid ()                            */
/*                             DrawPyramid ()                            */
/*************************************************************************/

#include <starbase.c.h>
#include <stdio.h>
#include <math.h>

#include "whole.h"

void drawline();
void drawsphere();
void ResetView ();
int  xt, yt, zt;

/*************************************************************************/
/*   Function Name:   newdip ()                                          */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Display menus on the screen                                */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function displays three menus one by one on the screen, and */
/*   provides choices for x coordinate, y coordinate and z coordinate    */
/*   respectively.                                                       */
/*************************************************************************/

void newdip()
```

```
{
   void    PutMenu(), ShowChoice();
   void    new_cube(), new_pyramid();
   void    RealCube();
   Boolean TakeChoiceNew(), TakeXMenuNew(), TakeYMenuNew(), TakeZMenuNew();
   MenuSt  LoadMenu(), *GetChoice();
   int     Choice, OldChoice = NO_CHOICE;
   MenuSt  XMenu, YMenu, ZMenu, NEWMenu;
   char    MenuFileName[80];
   Boolean MenuDone, XMenuDone, YMenuDone, ZMenuDone;


     strcpy(MenuFileName, "XMenu.menu");
     GET_PATH (MENU_PATH, MenuFileName);
     XMenu = LoadMenu (MenuFileName);
     DelAllMesg (Overlay);
     PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - XMenu.Width, 0, &XMenu);
     do {
        while (GetChoice (Mouse, Overlay, &Choice) != &XMenu);
        ShowChoice (Overlay, Choice,OldChoice,  &XMenu);
        DelAllMesg (Overlay);
        OldChoice = Choice;
        XMenuDone = TakeXMenuNew (Choice);
        if (XMenuDone) {
            PutMenu (Overlay, NULL_STRING, RIGHT_EDGE-XMenu.Width, 0, &XMenu);
            ShowChoice (Overlay, Choice, NO_CHOICE, &XMenu);
        }
     } while (Choice != XMenu.NbrEnts);
      ClearScreen (Overlay);

      strcpy (MenuFileName, "YMenu.menu");
      GET_PATH (MENU_PATH, MenuFileName);
      YMenu = LoadMenu (MenuFileName);
      DelAllMesg (Overlay);
      OldChoice = NO_CHOICE;
      PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - YMenu.Width, 0, &YMenu);
      do {
         while (GetChoice (Mouse, Overlay, &Choice) != &YMenu);
         ShowChoice (Overlay, Choice, OldChoice,  &YMenu);
         DelAllMesg (Overlay);
         OldChoice = Choice;
         YMenuDone = TakeYMenuNew (Choice);
         if (YMenuDone) {
             PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - YMenu.Width, 0, &YMenu);
             ShowChoice (Overlay, Choice, NO_CHOICE, &YMenu);
         }
       } while (Choice != YMenu.NbrEnts);
       ClearScreen (Overlay);

       strcpy (MenuFileName, "ZMenu.menu");
       GET_PATH (MENU_PATH, MenuFileName);
       ZMenu = LoadMenu (MenuFileName);
       DelAllMesg (Overlay);
       OldChoice = NO_CHOICE;
       PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - ZMenu.Width, 0, &ZMenu);
       do {
          while (GetChoice (Mouse, Overlay, &Choice) != &ZMenu);
          ShowChoice (Overlay, Choice, OldChoice,  &ZMenu);
          DelAllMesg (Overlay);
          OldChoice = Choice;
          ZMenuDone = TakeZMenuNew (Choice);
          if (ZMenuDone) {
              PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - ZMenu.Width, 0, &ZMenu);
              ShowChoice (Overlay, Choice, NO_CHOICE, &ZMenu);
          }
        } while (Choice != ZMenu.NbrEnts);
```

```
      ClearScreen (Overlay);

      strcpy (MenuFileName, "NEWMenu.menu");
      GET_PATH (MENU_PATH, MenuFileName);
      NEWMenu = LoadMenu (MenuFileName);
      DelAllMesg (Overlay);
      OldChoice = NO_CHOICE;
      PutMenu (Overlay, NULL_STRING, RIGHT_EDGE - NEWMenu.Width, 0, &NEWMenu);
      do
      {
         while (GetChoice (Mouse, Overlay, &Choice) != &NEWMenu);
         ShowChoice (Overlay, Choice,OldChoice, &NEWMenu);
         DelAllMesg (Overlay);
         OldChoice = Choice;
         MenuDone = TakeChoiceNew (Choice);
         if (MenuDone) {
            PutMenu (Overlay, NULL_STRING, RIGHT_EDGE-NEWMenu.Width,0, &NEWMenu);
            ShowChoice (Overlay, Choice, NO_CHOICE, &NEWMenu);
         }
       } while (Choice != NEWMenu.NbrEnts);
/* ResetView (Display); */
ClearScreen (Overlay);
/* ClearScreen (Display); */
RestoreActiveMenus (Overlay);
}

/*****************************************************************************/
/*   Function Name:   TakeChoiceNew ()                                       */
/*                                                                           */
/*   Input:   choice - A pointer to menu choice                              */
/*                                                                           */
/*   Output:  None                                                           */
/*                                                                           */
/*   Return:  Gone - A Boolean variable                                      */
/*                                                                           */
/*   Description:                                                            */
/*      This function provides the choices for the newdip.c, it includes     */
/*   using cube and pyramid models for local structure.                      */
/*****************************************************************************/

Boolean TakeChoiceNew (Choice)
{
   void new_cube(),   new_pyramid();
   void XaxisCube(), YaxisCube(), ZaxisCube();
   void XYaxisCube(), XZaxisCube(), YZaxisCube(),XYZaxisCube();
   void XPyramid(), YPyramid(), ZPyramid(), XYPyramid();
   void XZPyramid(), YZPyramid(), XYZPyramid();
   Boolean Gone = FALSE;

   switch (Choice) {
      case 1: Gone = TRUE;
              if ((xt<8) & (yt<8) & (zt<8))
                 new_cube();
              else if ((xt>8) & (yt<8) & (zt<8))
                     XaxisCube();
                     else if ((xt>8) & (yt>8) & (zt<8))
                        XYaxisCube();
                        else if ((xt>8) & (yt>8) & (zt>8))
                           XYZaxisCube();
                           else if ((xt<8) & (yt>8) & (zt<8))
                              YaxisCube();
                              else if ((xt<8) & (yt>8) & (zt>8))
                                 YZaxisCube();
                                 else if ((xt<8) & (yt<8) & (zt>8))
                                    ZaxisCube();
```

```
                                    else if ((xt>8) & (yt<8) & (zt>8))
                                       XZaxisCube();
                break;
        case 2: Gone = TRUE;
                if ((xt<8) & (yt<8) & (zt<8))
                    new_pyramid();
                    else if ((xt>8) & (yt<8) & (zt<8))
                         XPyramid();
                         else if ((xt>8) & (yt>8) & (zt<8))
                              XYPyramid();
                              else if ((xt>8) & (yt>8) & (zt>8))
                                   XYZPyramid();
                                   else if ((xt<8) & (yt>8) & (zt<8))
                                        YPyramid();
                                        else if ((xt<8) & (yt>8) & (zt>8))
                                             YZPyramid();
                                             else if ((xt>8) & (yt<8) & (zt>8))
                                                  XZPyramid();
                                                  else if ((xt<8) & (yt<8) & (zt>8))
                                                       ZPyramid();
                  break;
        case 3: break;
    }
    return (Gone);
}

/*****************************************************************************/
/*   Function Name:   TakeXMenuNew ()                                        */
/*                                                                           */
/*   Input:   Choice - A pointer to menu choice                              */
/*                                                                           */
/*   Output:  None                                                           */
/*                                                                           */
/*   Return:  Gone - A Boolean variable                                      */
/*                                                                           */
/*   Description:                                                            */
/*      This function provides the choices for the menu to x coordiante.     */
/*****************************************************************************/

Boolean TakeXMenuNew (Choice)
{
    Boolean Gone = FALSE;

    switch (Choice ) {
       case 1: Gone = TRUE; xt=0; break;
       case 2: Gone = TRUE; xt=1; break;
       case 3: Gone = TRUE; xt=2; break;
       case 4: Gone = TRUE; xt=3; break;
       case 5: Gone = TRUE; xt=4; break;
       case 6: Gone = TRUE; xt=5; break;
       case 7: Gone = TRUE; xt=6; break;
       case 8: Gone = TRUE; xt=7; break;
       case 9: Gone = TRUE; xt=999; break;
       case 10: break;
    }
    return (Gone);
}

/*****************************************************************************/
/*   Function Name:   TakeYMenu ()                                           */
/*                                                                           */
/*   Input:   Choice - A pointer to menu choice                              */
/*                                                                           */
/*   Output:  None                                                           */
/*                                                                           */
/*   Return:  Gone - A Boolean variable                                      */
```

```
/*                                                                          */
/*   Description:                                                           */
/*      This function provides the choices for the menu to y coordiante.    */
/****************************************************************************/

Boolean TakeYMenuNew (Choice)
{
    Boolean Gone = FALSE;

    switch (Choice) {
       case 1: Gone = TRUE; yt=0; break;
       case 2: Gone = TRUE; yt=1; break;
       case 3: Gone = TRUE; yt=2; break;
       case 4: Gone = TRUE; yt=3; break;
       case 5: Gone = TRUE; yt=4; break;
       case 6: Gone = TRUE; yt=5; break;
       case 7: Gone = TRUE; yt=6; break;
       case 8: Gone = TRUE; yt=7; break;
       case 9: Gone = TRUE; yt=999; break;
       case 10: break;
    }
    return (Gone);
}

/****************************************************************************/
/*   Function Name:   TakeZMenu ()                                          */
/*                                                                          */
/*   Input:   Choice - A pointer to menu choice                             */
/*                                                                          */
/*   Output:  None                                                          */
/*                                                                          */
/*   Return:  Gone - A Boolean variable                                     */
/*                                                                          */
/*   Description:                                                           */
/*      This function provides the choices for the menu to z coordinate.    */
/****************************************************************************/

Boolean TakeZMenuNew (Choice)
{
    Boolean Gone = FALSE;

    switch (Choice) {
       case 1: Gone = TRUE; zt=0; break;
       case 2: Gone = TRUE; zt=1; break;
       case 3: Gone = TRUE; zt=2; break;
       case 4: Gone = TRUE; zt=3; break;
       case 5: Gone = TRUE; zt=4; break;
       case 6: Gone = TRUE; zt=5; break;
       case 7: Gone = TRUE; zt=6; break;
       case 8: Gone = TRUE; zt=7; break;
       case 9: Gone = TRUE; zt=999; break;
       case 10: break;
    }
    return (Gone);
}

/****************************************************************************/
/*   Function Name:   draw_cubicNew ()                                      */
/*                                                                          */
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  Display a cube on the screen                                  */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
```

```
/*      This function draws a big cube as a border of model.                */
/*************************************************************************/

void draw_cubicNew()
{
   line_color_index (Display, 155);
   move3d (Display, 0.0, 0.0, 0.0);
   draw3d (Display, 0.0, 1.0, 0.0);
   draw3d (Display, 1.0, 1.0, 0.0);
   draw3d (Display, 1.0, 0.0, 0.0);
   draw3d (Display, 0.0, 0.0, 0.0);
   draw3d (Display, 0.0, 0.0, 1.0);
   draw3d (Display, 0.0, 1.0, 1.0);
   draw3d (Display, 1.0, 1.0, 1.0);
   draw3d (Display, 1.0, 0.0, 1.0);
   draw3d (Display, 0.0, 0.0, 1.0);
   move3d (Display, 0.0, 1.0, 0.0);
   draw3d (Display, 0.0, 1.0, 1.0);
   move3d (Display, 1.0, 1.0, 0.0);
   draw3d (Display, 1.0, 1.0, 1.0);
   move3d (Display, 1.0, 0.0, 0.0);
   draw3d (Display, 1.0, 0.0, 1.0);
}

/*************************************************************************/
/*   Function Name:   DrawSmCubicNew ()                                  */
/*                                                                       */
/*   Input:   xval1 - The relative coordinate of the first point on x    */
/*            yval1 - The relative coordinate of the first point on y    */
/*            zval1 - The relative coordiante of the first point on z    */
/*            xval2 - The relative coordiante of the second point on x   */
/*            yval2 - The relative coordiante of the second point on y   */
/*            zval2 - The relative coordinate of the second point on z   */
/*            ColArg - The value of color                                */
/*                                                                       */
/*   Output:  The small reference cube on the screen                     */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function draws a small reference cube as a cell for 12 voxel*/
/*   group.                                                              */
/*************************************************************************/

void DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,ColArg)
int ColArg;
float xval1,yval1,zval1,xval2,yval2,zval2;
{
   line_color_index (Display, ColArg);
   move3d (Display, xval1, yval1, zval1);
   draw3d (Display, xval1, yval2, zval1);
   draw3d (Display, xval2, yval2, zval1);
   draw3d (Display, xval2, yval1, zval1);
   draw3d (Display, xval1, yval1, zval1);
   draw3d (Display, xval1, yval1, zval2);
   draw3d (Display, xval1, yval2, zval2);
   draw3d (Display, xval2, yval2, zval2);
   draw3d (Display, xval2, yval1, zval2);
   draw3d (Display, xval1, yval1, zval2);
   move3d (Display, xval1, yval2, zval1);
   draw3d (Display, xval1, yval2, zval2);
   move3d (Display, xval2, yval2, zval1);
   draw3d (Display, xval2, yval2, zval2);
   move3d (Display, xval2, yval1, zval1);
   draw3d (Display, xval2, yval1, zval2);
}
```

```
/*************************************************************************/
/*   Function Name:   new_cube ()                                        */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Reference cube and cube model for voxels in a cell         */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function calls DrawSmCubicNew to draw a reference cube and  */
/*   draws 12 small cubes for voxels in a cell.                          */
/*************************************************************************/

void new_cube()
{
   void draw_cubicNew();
   void DrawSmCubicNew();
   void RealLinkNew ();
   int Buffer=0;
   float xval1,yval1,zval1,xval2,yval2,zval2;

      xrefpt = (float)xt /8.0;
      yrefpt = (float)yt /8.0;
      zrefpt = (float)zt /8.0;
      ClearScreen(Display);
      hidden_surface (Display, HSRoff, CULLoff);
      double_buffer (Display,TRUE, 8);

      for (Theta=0.0; Theta<360.1; Theta+=2.0) {
          Camera.camx = 3*cos((Theta+50) deg)+0.5;
          Camera.camz = 3.5;
          Camera.camy = 3*sin((Theta+50) deg)+0.5;
          view_camera (Display, &Camera);
          dbuffer_switch (Display, Buffer = !Buffer);
      draw_cubicNew ();
       /* Tics (Display, 'X', 0.16, 0.0, 1.0, 1.0, 0.0002, 5);  */
      xval1 = xt*0.125;
      yval1 = yt*0.125;
      zval1 = zt*0.125;
      xval2 = (1+xt)*0.125;
      yval2 = (1+yt)*0.125;
      zval2 = (1+zt)*0.125;
      DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
      RealLinkNew (xt,yt,zt);
      }
      make_picture_current(Display);
}

/*************************************************************************/
/*   Function Name:   RealLinkNew ()                                     */
/*                                                                       */
/*   Input:   xt - The x coordinate of parameter                         */
/*            yt - The y coordiante of parameter                         */
/*            zt - The z coordiante of parameter                         */
/*                                                                       */
/*   Output:  Cube model for voxel                                       */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function calls DrawSmCubicNew to draw a small cube as the   */
/*   model for voxels in the new_cube ().                                */
/*************************************************************************/
```

```
void RealLinkNew(xt,yt,zt)
int xt,yt,zt;
{
   int i;
   void DrawSmCubicNew();
   float xval1,yval1,zval1,xval2,yval2,zval2;

      for (i=0; i < 8; i++) {
         CaseArg =  prot[xt][yt][zt][i] + 200;
         xval1 = xrefpt + xshift[i];
         yval1 = yrefpt + yshift[i];
         zval1 = zrefpt + zshift[i];
         xval2 = xrefpt + xshift[i] + 0.010;
         yval2 = yrefpt + yshift[i] + 0.010;
         zval2 = zrefpt + zshift[i] + 0.010;
         DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,CaseArg);
  }
      for (i=0; i < 4; i++) {
         CaseArg = alka[xt][yt][zt][i] + 150;
         xval1 = xrefpt + xshift[i+8];
         yval1 = yrefpt + yshift[i+8];
         zval1 = zrefpt + zshift[i+8];
         xval2 = xrefpt + xshift[i+8] + 0.010;
         yval2 = yrefpt + yshift[i+8] + 0.010;
         zval2 = zrefpt + zshift[i+8] + 0.010;
         DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,CaseArg);
     }
}

/*************************************************************************/
/*   Function Name:   XaxisCube ()                                       */
/*                                                                       */
/*   Input:   None                                                       */
/*                                                                       */
/*   Output:  Slice cube and 8 * 12 cubes for voxels along x coordiante  */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function calls DrawSmCubeNew to draw a reference slice cube */
/*   and draws 8 * 12 cube in 8 cells along x coordiante.                */
/*************************************************************************/

void XaxisCube ()
{
    void draw_cubicNew();
    void RealLinkNew();
    void DrawSmCubicNew();
    int x, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen (Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE,8);
    for (Theta = 0.0; Theta < 360.1; Theta += 5.0) {
        Camera.camx = 3*cos((Theta+50) deg)+0.5;
        Camera.camz = 3.5;
        Camera.camy = 3*sin((Theta+50) deg)+0.5;
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubicNew ();
    xval1 = 0.0;
    yval1 = yt*0.125;
    zval1 = zt*0.125;
    xval2 = 1.0;
```

```
    yval2 = (1+yt)*0.125;
    zval2 = (1+zt)*0.125;
    DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);

    yrefpt = (float)yt /8.0;
    zrefpt = (float)zt /8.0;
    for (x=0; x<8; x++) {
       xrefpt = (float)x /8.0;
       RealLinkNew (x, yt, zt);
    }
   }
    make_picture_current (Display);
}

/*************************************************************************/
/*   Function  Name:   YaxisCube ()                                      */
/*                                                                       */
/*   Input:    None                                                      */
/*                                                                       */
/*   Output:  Slice cube and 8 * 12 cubes for voxels along y coordainte  */
/*                                                                       */
/*   Return:  None                                                       */
/*                                                                       */
/*   Description:                                                        */
/*      This function calls DrawSmCubicNew to draw a reference slice cube*/
/*   and draws 8 * 12 cubes in 8 cells along y coordinate.               */
/*************************************************************************/

void YaxisCube ()
{
    void draw_cubicNew();
    void RealLinkNew();
    void DrawSmCubicNew();

    int y, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen(Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE, 8);
    for (Theta =0.0; Theta < 360.1; Theta += 5.0) {
        Camera.camx = 3*cos((Theta+50) deg)+0.5;
        Camera.camz = 3.5;
        Camera.camy = 3*sin((Theta+50) deg)+0.5;

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);

    draw_cubicNew ();
    xval1 = xt*0.125;
    yval1 = 0.0;
    zval1 = zt*0.125;
    xval2 = (1+xt)*0.125;
    yval2 = 1.0;
    zval2 = (1+zt)*0.125;
    DrawSmCubicNew (xval1, yval1, zval1, xval2, yval2, zval2,79);

    xrefpt = (float)xt /8.0;
    zrefpt = (float)zt /8.0;
    for (y=0; y<8; y++) {
       yrefpt = (float)y /8.0;
       RealLinkNew (xt, y, zt);
    }
   }
    make_picture_current (Display);
}
```

```
/*****************************************************************************/
/*   Function Name:   ZaxisCube ()                                           */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice cube and 8 * 12 cubes for voxels along z coordiante      */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubeNew to draw a reference slice cube     */
/*   and draws 8 * 12 cubes in 8 cells along z coordiante.                   */
/*****************************************************************************/

void ZaxisCube ()
{
    void draw_cubicNew();
    void RealLinkNew();
    void DrawSmCubicNew();
    int z, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen(Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE, 8);
    for (Theta = 0.0; Theta < 360.1; Theta += 5.0) {
        Camera.camx = 3*cos((Theta+50) deg)+0.5;
        Camera.camz = 3.5;
        Camera.camy = 3*sin((Theta+50) deg)+0.5;

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubicNew();
    xval1 = xt*0.125;
    yval1 = yt*0.125;
    zval1 = 0.0;
    xval2 = (1+xt)*0.125;
    yval2 = (1+yt)*0.125;
    zval2 = 1.0;
    DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);

    xrefpt = (float)xt /8.0;
    yrefpt = (float)yt /8.0;
    for (z=0; z<8; z++) {
       zrefpt = (float)z /8.0;
       RealLinkNew (xt, yt, z);
    }
   }
    make_picture_current (Display);
}

/*****************************************************************************/
/*   Fountion Name:   XYaxisCube ()                                          */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 cubes along x-y coordiante          */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference slice plane   */
/*   and draws 8 * 8 * 12 cubes in 8 * 8 cells along x-y coordiante.         */
/*****************************************:***********************************/
```

```
void XYaxisCube ()
{
    void draw_cubicNew();
    void RealLinkNew();
    void DrawSmCubicNew();
    int x,y, Buffer=0;
    float xval1,yval1,zval1,xval2,yval2,zval2;

    ClearScreen (Display);
    hidden_surface (Display, HSRoff, CULLoff);
    double_buffer (Display, TRUE, 8);
    for (Theta=0.0; Theta<360.1; Theta+=8.0) {
        Camera.camx = 3*cos((Theta+50) deg)+0.5;
        Camera.camz = 3.5;
        Camera.camy = 3*sin((Theta+50) deg)+0.5;

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
    draw_cubicNew();
    xval1 = 0.0;
    yval1 = 0.0;
    zval1 = zt*0.125;
    xval2 = 1.0;
    yval2 = 1.0;
    zval2 = (1+zt)*0.125;
    DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);

    zrefpt = (float)zt /8.0;
    for (x=0; x<8; x++) {
       xrefpt = (float)x /8.0;
       for (y=0; y<8; y++) {
          yrefpt = (float)y /8.0;
          RealLinkNew (x, y, zt);
       }
    }
   }
    make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   XZaxisCube ()                                          */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 cubes along x-z coordiante          */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference slice plane   */
/*   and draws 8 * 8 * 12 cubes in 8 * 8 cells along x-z coordiante.         */
/*****************************************************************************/

void XZaxisCube ()
{
     void draw_cubicNew();
     void RealLinkNew();
     void DrawSmCubicNew();
     int x, z,Buffer=0;
     float xval1,yval1,zval1,xval2,yval2,zval2;

     ClearScreen(Display);
     hidden_surface (Display, HSRoff, CULLoff);
     double_buffer (Display, TRUE, 8);
     for (Theta=0.0; Theta<360.1; Theta+=8.0) {
         Camera.camx = 3*cos((Theta+50) deg)+0.5;
```

```
        Camera.camz = 3.5;
        Camera.camy = 3*sin((Theta+50) deg)+0.5;

        view_camera(Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
     draw_cubicNew();
     xval1 = 0.0;
     yval1 = yt*0.125;
     zval1 = 0.0;
     xval2 = 1.0;
     yval2 = (1+yt)*0.125;
     zval2 = 1.0;
     DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);

     yrefpt = (float)yt /8.0;
     for (x=0; x<8; x++) {
        xrefpt = (float)x /8.0;
        for (z=0; z<8; z++) {
           zrefpt = (float)z /8.0;
           RealLinkNew (x, yt, z);
        }
     }
    }
     make_picture_current(Display);
}

/*****************************************************************************/
/*   Function Name:   YZaxisCube ()                                          */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 cubes along y-z coordiante          */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference slice plane   */
/*   and draws 8 * 8 * 12 cubes in 8 * 8 cells along y-z coordiante.         */
/*****************************************************************************/

void YZaxisCube ()
{
     void draw_cubicNew();
     void RealLinkNew();
     void DrawSmCubicNew();
     int y, z, Buffer=0;
     float xval1,yval1,zval1,xval2,yval2,zval2;

     ClearScreen (Display);
     hidden_surface(Display, HSRoff, CULLoff);
     double_buffer(Display, TRUE, 8);
     for (Theta=0.0; Theta<360.1; Theta+=8.0) {
         Camera.camx = 3*cos((Theta+50) deg)+0.5;
         Camera.camz= 3.5;
         Camera.camy = 3*sin((Theta+50) deg)+0.5;

         view_camera (Display, &Camera);
         dbuffer_switch (Display, Buffer = !Buffer);
     draw_cubicNew();
     xval1 = xt*0.125;
     yval1 = 0.0;
     zval1 = 0.0;
     xval2 = (1+xt)*0.125;
     yval2 = 1.0;
     zval2 = 1.0;
     DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
```

```
      xrefpt = (float)xt/8.0;
      for (y=0; y<8; y++) {
          yrefpt = (float)y /8.0;
          for (z=0; z<8; z++) {
              zrefpt = (float)z /8.0;
          RealLinkNew (xt, y, z);
          }
      }
    }
      make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   XYZaxisCube ()                                         */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  None                                                           */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function does nothing but returns to main menu.                */
/*****************************************************************************/

void XYZaxisCube ()
{
   return;
}

/*****************************************************************************/
/*   Function Name:   new-pyramid ()                                         */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Reference cube and pyramid model for voxels                    */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference cube and      */
/*   calls RealPyramid to draw a pyramid for voxel.                          */
/*****************************************************************************/

void new_pyramid()
{
   static Boolean  Col;
   void draw_cubicNew ();
   void RealPyramid();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   void DrawSmCubicNew();
   int Buffer = 0;

   ClearScreen (Display);

   hidden_surface (Display, HSRoff, CULLoff);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=2.0) {
            Camera.camx = 3*cos((Theta + 50) deg);
            Camera.camz = 5.0;
            Camera.camy = 3*sin((Theta + 50) deg);
         view_camera (Display, &Camera);
         dbuffer_switch (Display, Buffer = !Buffer);
         draw_cubicNew();
         xval1 = xt*0.125;
```

```
            yval1 = yt*0.125;
            zval1 = zt*0.125;
            xval2 = (1+xt)*0.125;
            yval2 = (1+yt)*0.125;
            zval2 = (1+zt)*0.125;
            DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
            xrefpt = (float)xt /8.0;
            yrefpt = (float)yt /8.0;
            zrefpt = (float)zt /8.0;
            RealPyramid (xt,yt,zt);
        }
        make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   XPyramid ()                                            */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Reference cube and 8 * 12 pyramids along x coordinate          */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference cube          */
/*   and 8 * 12 pyramids in 8 cells along x coordiante.                      */
/*****************************************************************************/

void XPyramid()
{
   void draw_cubicNew ();
   void RealPyramid ();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   void DrawSmCubicNew();
   int x, Buffer = 0;

   ClearScreen (Display);

   hidden_surface (Display, HSRoff, CULLoff);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=5.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
        draw_cubicNew();
        xval1 = 0.0;
        yval1 = yt*0.125;
        zval1 = zt*0.125;
        xval2 = 1.0;
        yval2 = (1+yt)*0.125;
        zval2 = (1+zt)*0.125;
        DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
        yrefpt = (float)yt/8.0;
        zrefpt = (float)zt/8.0;
        for (x=0; x<8; x++) {
            xrefpt = (float)x/8.0;
            RealPyramid(x,yt,zt);
    }
  }
    make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   YPyramid ()                                            */
```

```
/*                                                                          */
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  Reference cube and 8 * 12 pyramids along y coordiante         */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function calls DrawSmCubicNew to draw a reference cube         */
/*   and draws 8 * 12 pyramids in 8 cells along y coordiante.               */
/****************************************************************************/

void YPyramid()
{
   void draw_cubicNew ();
   void RealPyramid ();
   void DrawSmCubicNew();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int y, Buffer = 0;

   ClearScreen (Display);
   hidden_surface (Display, HSRoff, CULLoff);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+=5.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer =! Buffer);
        draw_cubicNew();
        xval1 = xt*0.125;
        yval1 = 0.0;
        zval1 = zt*0.125;
        xval2= (1+xt)*0.125;
        yval2 = 1.0;
        zval2 = (1+zt)*0.125;
        DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
        xrefpt = (float)xt/8.0;
        zrefpt = (float)zt/8.0;
        for (y=0; y<8; y++) {
            yrefpt = (float)y/8.0;
            RealPyramid(xt,y,zt);
       }
   }
       make_picture_current (Display);
}

/****************************************************************************/
/*   Function Name:   ZPyramid ()                                           */
/*                                                                          */
/*   Input:   None                                                          */
/*                                                                          */
/*   Output:  Reference cube and 8 * 12 pyramids along z coordiante         */
/*                                                                          */
/*   Return:  None                                                          */
/*                                                                          */
/*   Description:                                                           */
/*      This function calls DrawSmCubicNew to draw a reference cube         */
/*   and draws 8 * 12 pyramids in 8 cells along z coordiante.               */
/****************************************************************************/

void ZPyramid ()
{
     void draw_cubicNew();
     void DrwmSmCubic();
     void RealPyramid();
```

```
        float xval1,yval1,zval1,xval2,yval2,zval2;
        int z, Buffer = 0;

        ClearScreen (Display);
        hidden_surface (Display, HSRoff, CULLoff);
        double_buffer (Display, TRUE, 8);
        for (Theta=0.0; Theta<360.1; Theta+=5.0) {
             Camera.camx = 3*cos((Theta+50) deg);
             Camera.camz = 5.0;
             Camera.camy = 3*sin((Theta+50) deg);
             view_camera (Display, &Camera);
             dbuffer_switch (Display, Buffer = !Buffer);
             draw_cubicNew();
             xval1 = xt*0.125;
             yval1 = yt*0.125;
             zval1 = 0.0;
             xval2 = (1+xt)*0.125;
             yval2 = (1+yt)*0.125;
             zval2 = 1.0;
             DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
             xrefpt = (float)xt/8.0;
             yrefpt = (float)yt/8.0;
             for (z=0.0; z<8; z++) {
                 zrefpt = (float)z/8.0;
                 RealPyramid (xt,yt,z);
             }
          }
             make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   XYPyramid ()                                           */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 pyramid along x-y coordiante        */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference slice plane   */
/*   and draws 8 * 8 * 12 pyramids in 8 * 8 cells along x-y coordiante.      */
/*****************************************************************************/

void XYPyramid ()
{
   void draw_cubicNew();
   void DrawSmCubicNew ();
   void RealPyramid();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int x,y, Buffer = 0;

   ClearScreen (Display);
   hidden_surface (Display, HSRoff, CULLoff);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+= 8.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
        draw_cubicNew();
        xval1 =0.0;
        yval1 = 0.0;
        zval1 = zt*0.125;
        xval2 = 1.0;
```

```
        yval2 = 1.0;
        zval2 = (1+zt)*0.125;
        DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
        zrefpt = (float)zt/8.0;
        for (x=0; x<8; x++) {
            xrefpt = (float)x/8.0;
            for (y=0; y<8; y++) {
                yrefpt = (float)y/8.0;
                RealPyramid (x,y,zt);
            }
         }
    }
         make_picture_current(Display);
}

/*****************************************************************************/
/*   Function Name:   XZPyramid ()                                           */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 pyramids along x-z coordiante       */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference slice plane   */
/*   and draws 8 * 8 * 12 pyramids in 8 * 8 cells along x-z coordiante.      */
/*****************************************************************************/

void XZPyramid()
{
   void draw_cubicNew();
   void RealPyramid();
   void DrawSmCubicNew();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int x,z, Buffer = 0;

   ClearScreen (Display);
   hidden_surface (Display, HSRoff, CULLoff);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+= 8.0) {
        Camera.camx = 3*cos((Theta+50) deg);
        Camera.camz = 5.0;
        Camera.camy = 3*sin((Theta+50) deg);
        view_camera (Display, &Camera);
        dbuffer_switch (Display, Buffer = !Buffer);
        draw_cubicNew();
        xval1 = 0.0;
        yval1 = yt*0.125;
        zval1 = 0.0;
        xval2 = 1.0;
        yval2 = (1+yt)*0.125;
        zval2 = 1.0;
        DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
        yrefpt = (float)yt/8.0;
        for (x=0; x<8; x++) {
            xrefpt = (float)x/8.0;
            for (z=0; z<8; z++) {
                 zrefpt = (float)z/8.0;
                 RealPyramid(x,yt,z);
            }
        }
    }
    make_picture_current(Display);
}
```

```
/*****************************************************************************/
/*   Function Name:   YZPyramid ()                                           */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  Slice plane and 8 * 8 * 12 pyramids along y-z coordinate       */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function calls DrawSmCubicNew to draw a reference slice plane   */
/*   and draws 8 * 8 * 12 pyramids in 8 * 8 cells along y-z coordiante.      */
/*****************************************************************************/

void YZPyramid()
{
   void draw_cubicNew();
   void RealPyramid();
   void DrawSmCubicNew();
   float xval1,yval1,zval1,xval2,yval2,zval2;
   int y,z, Buffer = 0;

   ClearScreen(Display);
   hidden_surface (Display, HSRoff, CULLoff);
   double_buffer (Display, TRUE, 8);
   for (Theta=0.0; Theta<360.1; Theta+= 8.0) {
       Camera.camx = 3*cos((Theta+50) deg);
       Camera.camz = 5.0;
       Camera.camy = 3*sin((Theta+50) deg);
       view_camera (Display, &Camera);
       dbuffer_switch (Display, Buffer = !Buffer);
       draw_cubicNew();
       xval1 = xt*0.125;
       yval1 = 0.0;
       zval1 = 0.0;
       xval2 = (1+xt)*0.125;
       yval2 = 1.0;
       zval2 = 1.0;
       DrawSmCubicNew (xval1,yval1,zval1,xval2,yval2,zval2,79);
       xrefpt = (float)xt/8.0;
       for (y=0; y<8; y++) {
           yrefpt = (float)y/8.0;
           for (z=0; z<8; z++) {
               zrefpt = (float)z/8.0;
               RealPyramid(xt,y,z);
           }
       }
     }
     make_picture_current (Display);
}

/*****************************************************************************/
/*   Function Name:   XYZPyramid ()                                          */
/*                                                                           */
/*   Input:   None                                                           */
/*                                                                           */
/*   Output:  None                                                           */
/*                                                                           */
/*   Return:  None                                                           */
/*                                                                           */
/*   Description:                                                            */
/*      This function does nothing but returns to main menu.                */
/*****************************************************************************/

void XYZPyramid ()
{
```

```
   return;
}

/*****************************************************************+***************/
/*   Function Name:   RealPyramid ()                                              */
/*                                                                                */
/*   Input:   xt - The x coordinate of parameter                                  */
/*            yt - The y coordinate of parameter                                  */
/*            zt - The z coordinate of parameter                                  */
/*                                                                                */
/*   Output:  None                                                                */
/*                                                                                */
/*   Return:  None                                                                */
/*                                                                                */
/*   Description:                                                                 */
/*      This function provides the coordiantes for DrawPyramid to draw            */
/*   pyramid.                                                                     */
/**********************************************************************************/

void RealPyramid(xt,yt,zt)
int xt,yt,zt;
{
   int Col;
   int i;
   float xval1,yval1,zval1,xval2,yval2,zval2,xval3,yval3,zval3;
   void DrawPyramid();

           for (i=0; i<8; i++) {
              Col = prot[xt][yt][zt][i] + 200;
              xval1 = xrefpt + xshift[i];
              yval1 = yrefpt + yshift[i];
              zval1 = zrefpt + zshift[i];
              xval2 = xrefpt + xshift[i] + 0.005;
              yval2 = yrefpt + yshift[i] + 0.005;
              zval2 = zrefpt + zshift[i] + 0.005;
              xval3 = xrefpt + xshift[i] + 0.010;
              yval3 = yrefpt + yshift[i] + 0.010;
              zval3 = zrefpt + zshift[i] + 0.010;
    DrawPyramid(xval1,yval1,zval1,xval2,yval2,zval2,xval3,yval3,zval3,Col);
           }
           for (i=0; i<4; i++) {
              Col = alka[xt][yt][zt][i] + 150;
              xval1 = xrefpt + xshift[i+8];
              yval1 = yrefpt + yshift[i+8];
              zval1 = zrefpt + zshift[i+8];
              xval2 = xrefpt + xshift[i+8] + 0.005;
              yval2 = yrefpt + yshift[i+8] + 0.005;
              zval2 = zrefpt + zshift[i+8] + 0.005;
              xval3 = xrefpt + xshift[i+8] + 0.010;
              yval3 = yrefpt + yshift[i+8] + 0.010;
              zval3 = zrefpt + zshift[i+8] + 0.010;
    DrawPyramid(xval1,yval1,zval1,xval2,yval2,zval2,xval3,yval3,zval3,Col);
           }
}

/**********************************************************************************/
/*   Function Name:   DrawPyramid ()                                              */
/*                                                                                */
/*   Input:   xval1 - The x coordinate of the first reference point               */
/*            yval1 - The y coordinate of the first reference point               */
/*            zval1 - The z coordinate of the first reference point               */
/*            xval2 - The x coordinate of the second reference point              */
/*            yval2 - The y coordinate of the second reference point              */
/*            zval2 - The z coordinate of the second reference point              */
/*            xval3 - The x coordinate of the third reference point               */
/*            yval3 - The y coordinate of the third reference point               */
```

```
/*            zval3 - The z coordinate of the third reference point        */
/*            ColArg - The value of color                                  */
/*                                                                         */
/*   Output:  Pyramid                                                      */
/*                                                                         */
/*   Return:  None                                                         */
/*                                                                         */
/*   Description:                                                          */
/*      This function draws a pyramid as the model for voxel.              */
/*************************************************************************/
void DrawPyramid(xval1,yval1,zval1,xval2,yval2,zval2,xval3,yval3,zval3,ColArg)
int ColArg;
float xval1,yval1,zval1,xval2,yval2,zval2,xval3,yval3,zval3;
{
   line_color_index (Display, ColArg);
   move3d (Display, xval1,yval1,zval1);
   draw3d (Display, xval2,yval2,zval2);
   draw3d (Display, xval3,yval1,zval1);
   draw3d (Display, xval1,yval1,zval1);
   draw3d (Display, xval1,yval1,zval2);
   draw3d (Display, xval2,yval2,zval2);
   draw3d (Display, xval3,yval1,zval3);
   draw3d (Display, xval3,yval1,zval1);
   draw3d (Display, xval3,yval1,zval3);
   draw3d (Display, xval1,yval1,zval3);
}
```

Figure 5. INPUT DATA SAMPLE

```
-1  1 -1 -1 -1 -1 -1 -1    -1 -1 -1  1
-1 -1 -1  1 -1 -1 -1 -1     1 -1 -1 -1
-1 -1 -1  1  1 -1 -1  1     1  1 -1 -1
 1  1  1 -1  1 -1 -1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1  1 -1  1    -1 -1 -1 -1
-1 -1 -1  1  1 -1  1 -1    -1 -1 -1 -1
 1 -1 -1  1  1 -1 -1  1    -1 -1 -1 -1
 1  1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
-1  1  1  1  1  1 -1  1     1 -1 -1 -1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1  1 -1  1  1 -1 -1  1    -1  1 -1 -1
 1 -1 -1  1 -1  1  1  1     1 -1  1 -1
-1  1  1  1  1  1  1 -1    -1 -1  1 -1
 1  1 -1 -1 -1 -1 -1  1     1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1     1 -1 -1 -1
 1 -1  1  1  1  1  1  1     1 -1  1 -1
-1  1 -1  1  1 -1 -1 -1    -1  1  1 -1
-1 -1 -1 -1 -1 -1 -1  1     1 -1 -1 -1
 1 -1  1  1  1  1  1 -1     1  1 -1  1
 1 -1 -1  1  1 -1  1 -1    -1 -1 -1 -1
 1  1  1 -1 -1  1 -1  1    -1 -1 -1 -1
 1 -1  1  1  1  1  1 -1    -1  1 -1 -1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1 -1 -1  1 -1  1  1  1    -1 -1  1 -1
 1  1  1 -1  1 -1 -1  1     1 -1 -1 -1
-1  1  1  1  1  1 -1 -1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1  1    -1 -1 -1  1
-1 -1 -1  1 -1  1  1  1     1  1 -1 -1
 1  1  1 -1 -1  1  1 -1    -1  1 -1 -1
 1 -1  1  1  1 -1 -1  1    -1 -1 -1 -1
 1  1  1 -1  1  1 -1  1     1 -1 -1 -1
-1 -1 -1  1  1 -1 -1 -1     1 -1 -1  1
 1  1  1 -1  1 -1 -1 -1    -1 -1  1 -1
-1  1  1 -1 -1  1  1  1    -1 -1 -1  1
-1  1  1  1  1  1  1  1    -1  1 -1 -1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
 1 -1  1 -1  1 -1 -1 -1     1 -1 -1 -1
 1 -1  1  1  1  1  1  1    -1 -1  1 -1
 1 -1 -1  1  1 -1  1 -1    -1 -1  1  1
-1  1 -1  1  1 -1 -1  1    -1  1  1  1
 1 -1 -1 -1 -1 -1  1 -1     1 -1 -1  1
 1  1  1 -1  1 -1 -1 -1     1 -1 -1  1
 1 -1  1  1  1  1  1  1    -1  1  1 -1
-1  1 -1  1  1 -1 -1 -1     1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1 -1  1 -1
 1  1  1  1  1  1  1  1     1 -1 -1 -1
 1  1  1 -1  1 -1 -1  1    -1 -1 -1  1
 1 -1  1 -1  1 -1 -1  1    -1  1  1 -1
-1  1  1  1  1  1  1 -1    -1  1 -1 -1
```

```
 1 -1  1 -1 -1  1 -1 -1     1 -1  1 -1
-1 -1 -1  1  1  1  1 -1    -1 -1  1 -1
 1 -1 -1 -1 -1 -1 -1  1    -1  1 -1  1
-1 -1 -1 -1 -1 -1 -1  1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
 1 -1  1 -1 -1  1  1 -1    -1 -1 -1 -1
 1 -1  1 -1 -1 -1 -1  1     1 -1 -1 -1
 1 -1  1 -1  1 -1 -1  1     1  1 -1 -1
 1 -1  1 -1 -1  1  1 -1     1 -1 -1 -1
 1 -1  1  1  1  1  1  1    -1 -1  1 -1
 1  1  1 -1 -1 -1 -1 -1     1 -1 -1  1
-1 -1 -1  1  1  1 -1  1    -1 -1 -1 -1
 1 -1 -1  1 -1  1  1 -1    -1 -1  1 -1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1  1  1
-1 -1 -1  1  1 -1 -1 -1    -1  1 -1 -1
 1 -1 -1 -1 -1 -1  1 -1    -1  1 -1 -1
 1 -1 -1 -1 -1 -1 -1  1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1     1 -1 -1 -1
-1  1 -1 -1 -1 -1  1 -1     1  1  1 -1
-1  1  1  1  1  1  1  1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1 -1    -1  1  1 -1
-1 -1  1 -1  1 -1 -1 -1     1 -1 -1 -1
-1  1  1 -1 -1  1  1  1     1 -1 -1 -1
-1  1  1  1  1  1  1 -1    -1 -1  1  1
 1 -1 -1  1 -1  1  1  1    -1  1 -1  1
 1 -1 -1 -1 -1 -1 -1 -1     1  1 -1  1
-1 -1 -1  1 -1  1 -1  1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1 -1    -1  1 -1 -1
-1 -1 -1  1  1 -1 -1  1    -1 -1  1 -1
-1  1 -1  1 -1  1  1  1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1  1 -1 -1
-1  1  1  1 -1  1  1  1    -1 -1  1 -1
 1 -1 -1  1 -1  1  1  1    -1  1 -1  1
 1 -1 -1 -1 -1 -1 -1 -1     1  1 -1  1
-1 -1 -1  1 -1  1 -1  1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1 -1    -1 -1 -1 -1
-1  1 -1  1 -1  1 -1  1    -1  1  1 -1
-1  1 -1  1 -1  1 -1  1    -1  1  1 -1
 1  1  1  1  1  1 -1  1    -1 -1  1 -1
 1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1  1
 1 -1  1  1  1  1 -1  1    -1 -1 -1  1
 1 -1 -1  1  1 -1 -1 -1    -1 -1  1 -1
-1  1 -1 -1 -1 -1 -1  1    -1 -1 -1 -1
-1 -1 -1  1 -1  1  1  1    -1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1  1 -1  1
 1  1  1 -1  1 -1 -1 -1    -1 -1 -1  1
-1  1  1 -1 -1 -1 -1  1    -1 -1  1 -1
-1 -1 -1  1 -1  1 -1 -1     1 -1  1 -1
 1  1  1  1  1  1  1  1    -1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1 -1  1 -1
 1  1 -1 -1 -1 -1 -1 -1     1  1 -1 -1
 1 -1  1  1  1  1 -1  1     1 -1 -1 -1
-1  1 -1 -1 -1 -1  1 -1    -1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1 -1
-1  1  1  1  1  1  1  1    -1 -1  1 -1
 1 -1 -1  1  1 -1 -1  1     1 -1  1 -1
-1  1 -1 -1 -1 -1  1 -1     1 -1 -1  1
 1  1  1 -1  1 -1  1 -1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1  1  1 -1 -1  1  1  1    -1 -1  1  1
-1  1 -1  1 -1  1  1  1    -1 -1 -1 -1
-1  1 -1  1 -1 -1 -1 -1     1 -1  1 -1
```

```
 1  1  1  1  1  1  1  1     1 -1 -1 -1
 1  1  1 -1  1 -1  1 -1    -1  1  1 -1
-1 -1 -1 -1 -1 -1  1 -1    -1 -1  1 -1
-1  1  1 -1 -1  1  1  1    -1  1 -1 -1
-1 -1 -1 -1  1 -1 -1  1    -1  1  1 -1
-1 -1 -1  1 -1 -1  1 -1    -1 -1  1  1
-1  1 -1  1  1 -1 -1 -1    -1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1 -1  1  1
-1 -1 -1 -1 -1 -1  1 -1    -1  1 -1 -1
-1  1  1  1  1  1  1 -1     1 -1  1  1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1 -1     1 -1 -1 -1
-1  1  1  1  1  1  1  1    -1 -1 -1 -1
-1  1 -1  1 -1  1 -1  1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1  1     1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1 -1  1 -1
-1 -1 -1  1 -1  1  1 -1    -1 -1 -1  1
 1 -1 -1 -1 -1 -1 -1  1     1 -1 -1 -1
-1  1  1 -1 -1  1 -1  1    -1 -1 -1 -1
 1 -1  1 -1 -1  1  1 -1    -1 -1 -1  1
-1 -1 -1 -1 -1  1  1  1     1  1  1 -1
-1 -1 -1  1 -1  1 -1  1     1 -1 -1 -1
 1 -1 -1  1  1 -1  1 -1    -1  1  1 -1
 1  1  1  1  1  1 -1  1    -1  1 -1 -1
 1  1  1 -1 -1  1 -1  1    -1 -1 -1 -1
-1 -1 -1  1  1 -1 -1 -1    -1 -1 -1 -1
 1  1  1 -1  1 -1  1 -1     1  1  1 -1
 1 -1  1  1  1  1 -1  1    -1  1  1 -1
 1  1  1  1  1  1  1 -1     1 -1 -1 -1
 1 -1 -1 -1  1 -1 -1 -1     1  1 -1  1
 1  1  1 -1 -1 -1 -1  1    -1 -1 -1  1
-1 -1 -1 -1 -1 -1  1 -1     1 -1  1 -1
-1  1  1 -1  1 -1 -1 -1    -1 -1  1 -1
-1 -1 -1  1 -1  1 -1 -1    -1  1 -1 -1
 1 -1 -1 -1  1 -1 -1  1    -1 -1 -1 -1
-1  1  1 -1  1 -1 -1  1    -1 -1  1 -1
 1  1  1  1  1  1  1  1     1 -1  1 -1
-1 -1 -1  1 -1  1  1 -1    -1  1  1 -1
 1  1  1  1  1  1  1  1     1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1     1 -1  1 -1
-1  1  1  1  1  1 -1  1    -1 -1 -1 -1
 1 -1  1 -1  1 -1 -1  1     1 -1 -1  1
 1  1  1  1  1  1 -1  1    -1 -1 -1 -1
-1  1  1 -1  1 -1 -1  1    -1 -1 -1 -1
-1  1  1 -1 -1  1 -1  1    -1 -1 -1  1
 1 -1  1  1  1  1 -1 -1     1 -1  1 -1
 1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
 1 -1 -1  1  1  1  1 -1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1 -1
-1  1  1 -1  1 -1 -1  1    -1 -1  1 -1
 1 -1  1 -1  1 -1 -1  1    -1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1  1
 1 -1  1  1  1  1  1 -1     1  1  1  1
 1  1  1  1  1  1 -1  1    -1  1 -1 -1
 1  1  1  1  1  1 -1  1     1 -1 -1 -1
 1 -1  1 -1  1 -1 -1 -1     1  1 -1 -1
 1 -1  1 -1 -1  1  1 -1     1  1 -1 -1
 1 -1  1  1  1  1 -1  1    -1 -1  1  1
 1  1  1  1  1  1  1 -1    -1 -1 -1 -1
```

```
-1  1 -1  1 -1  1  1  1     1 -1 -1  1
 1 -1 -1 -1 -1 -1  1 -1    -1 -1 -1 -1
 1 -1  1  1  1 -1  1 -1    -1  1  1  1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1  1  1
-1  1  1  1  1  1  1  1    -1 -1 -1  1
 1 -1 -1 -1 -1 -1 -1 -1     1  1  1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1 -1  1 -1
-1  1  1 -1  1 -1  1 -1    -1 -1  1  1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1 -1 -1  1 -1  1 -1  1     1 -1 -1 -1
 1  1  1  1  1  1  1  1    -1 -1 -1  1
 1  1  1 -1  1 -1 -1 -1    -1 -1 -1  1
-1 -1 -1  1 -1  1  1 -1    -1 -1 -1  1
 1  1  1 -1 -1  1 -1  1     1  1 -1  1
-1 -1 -1  1 -1  1  1  1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1  1 -1    -1 -1 -1 -1
 1 -1  1  1 -1  1 -1  1     1  1 -1  1
-1 -1 -1  1 -1  1 -1  1    -1  1 -1 -1
-1 -1 -1  1 -1  1  1  1     1 -1 -1 -1
 1  1  1 -1 -1 -1  1 -1    -1  1 -1 -1
-1  1 -1  1  1  1  1 -1    -1  1 -1 -1
 1 -1 -1  1  1 -1 -1 -1    -1  1 -1 -1
 1  1  1 -1 -1  1  1 -1    -1 -1  1 -1
-1 -1  1 -1 -1  1 -1  1     1 -1  1 -1
-1 -1 -1  1 -1  1 -1  1     1  1 -1  1
 1  1  1  1  1  1 -1  1    -1 -1  1 -1
-1  1 -1  1 -1  1  1  1    -1  1 -1  1
-1  1 -1  1  1  1  1 -1    -1  1 -1 -1
 1 -1 -1 -1 -1 -1 -1  1    -1 -1  1  1
-1 -1 -1  1  1 -1 -1  1     1  1 -1  1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
 1  1  1 -1  1 -1 -1 -1    -1 -1 -1 -1
 1 -1  1  1  1  1  1  1    -1  1 -1 -1
 1 -1 -1  1 -1  1  1  1    -1 -1  1 -1
 1  1  1 -1  1 -1 -1 -1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1    -1  1 -1  1
-1 -1 -1  1 -1  1  1  1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1 -1     1 -1 -1  1
 1 -1  1  1  1  1  1  1    -1  1 -1 -1
 1 -1 -1 -1 -1  1 -1 -1    -1  1  1  1
-1  1 -1 -1 -1 -1 -1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1  1 -1 -1
-1  1  1  1  1  1  1  1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1 -1    -1 -1  1 -1
 1  1  1 -1 -1  1 -1  1     1 -1 -1  1
 1 -1  1 -1 -1  1  1  1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1  1 -1    -1  1 -1 -1
-1 -1 -1  1  1 -1  1 -1     1 -1  1 -1
 1  1  1  1  1  1  1 -1     1  1 -1 -1
-1  1 -1  1  1  1  1 -1     1  1 -1 -1
 1  1  1 -1  1 -1  1 -1    -1 -1 -1 -1
-1 -1 -1  1 -1  1  1  1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
-1  1  1  1  1  1 -1  1    -1 -1  1 -1
 1  1  1 -1  1 -1 -1 -1     1 -1 -1  1
-1  1  1  1  1  1  1  1     1 -1 -1 -1
 1  1  1 -1 -1  1  1  1    -1 -1 -1 -1
-1  1  1  1  1  1  1  1    -1 -1 -1 -1
```

```
 1 -1 -1  1  1 -1 -1 -1    -1 -1 -1 -1
 1  1  1 -1 -1  1  1  1    -1  1  1 -1
 1  1  1  1  1  1 -1  1     1 -1 -1 -1
-1  1 -1 -1 -1 -1 -1 -1     1  1 -1 -1
-1  1  1  1  1  1  1  1     1  1  1 -1
-1  1 -1  1 -1  1  1  1    -1  1 -1 -1
-1  1 -1  1 -1  1  1 -1    -1  1 -1  1
 1 -1 -1  1  1 -1  1 -1    -1 -1 -1 -1
 1  1  1 -1 -1  1  1  1    -1 -1  1 -1
-1  1  1  1  1  1  1 -1     1 -1 -1 -1
-1 -1 -1  1  1 -1  1 -1    -1 -1  1 -1
 1 -1 -1 -1 -1  1  1 -1    -1 -1  1 -1
-1 -1 -1 -1 -1 -1  1 -1    -1  1 -1 -1
-1  1  1  1  1  1 -1  1    -1 -1 -1 -1
 1 -1 -1  1 -1  1 -1  1     1  1  1 -1
 1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
-1 -1 -1 -1 -1 -1  1 -1    -1  1 -1 -1
-1  1  1  1  1  1  1  1    -1 -1 -1  1
 1  1  1 -1  1 -1 -1 -1    -1  1  1  1
-1 -1 -1  1  1 -1 -1 -1    -1  1 -1 -1
 1 -1 -1  1  1 -1 -1 -1    -1 -1 -1 -1
 1  1  1  1  1  1  1 -1    -1  1  1  1
 1  1  1 -1 -1  1 -1  1     1 -1  1 -1
 1 -1  1  1  1 -1 -1 -1    -1 -1 -1 -1
 1  1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
 1  1  1 -1  1 -1 -1  1    -1 -1 -1 -1
-1  1  1 -1 -1  1  1  1    -1 -1 -1 -1
 1 -1  1 -1  1 -1 -1 -1    -1 -1 -1 -1
-1  1 -1 -1 -1 -1  1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
-1  1 -1 -1  1 -1 -1  1    -1 -1 -1  1
-1 -1 -1 -1  1 -1 -1  1    -1 -1 -1  1
-1  1 -1  1 -1  1 -1  1     1 -1 -1 -1
 1 -1  1 -1  1 -1 -1  1     1  1 -1 -1
-1 -1 -1  1 -1  1  1  1    -1  1 -1 -1
 1  1  1 -1 -1  1  1  1    -1 -1  1 -1
-1 -1 -1  1 -1  1  1  1     1 -1  1 -1
 1 -1 -1  1  1 -1 -1  1    -1 -1  1 -1
 1  1  1  1 -1  1 -1  1     1  1 -1 -1
 1 -1  1  1  1 -1 -1  1     1 -1 -1 -1
 1 -1 -1 -1 -1 -1 -1 -1     1 -1 -1  1
 1  1 -1  1 -1  1 -1  1    -1  1 -1  1
-1 -1 -1  1 -1  1  1  1    -1  1 -1 -1
 1  1  1  1  1  1 -1  1    -1  1 -1  1
 1  1  1 -1  1 -1 -1 -1    -1  1 -1 -1
-1 -1 -1  1 -1  1  1  1    -1  1 -1 -1
-1  1 -1  1  1  1 -1  1     1  1 -1  1
 1 -1  1 -1  1 -1  1 -1    -1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1  1 -1  1
 1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1  1 -1 -1
-1  1  1  1  1  1 -1  1    -1 -1 -1  1
 1 -1 -1 -1 -1 -1 -1 -1     1 -1 -1 -1
 1 -1  1  1  1  1  1  1    -1  1  1  1
-1  1 -1  1  1 -1 -1 -1    -1  1  1 -1
 1 -1 -1  1 -1  1 -1  1    -1 -1  1 -1
```

```
 1 -1 -1  1 -1  1  1  1    -1 -1 -1 -1
-1 -1 -1  1 -1  1 -1  1    -1 -1 -1 -1
 1  1  1  1  1  1  1 -1     1 -1 -1 -1
 1  1  1  1  1  1  1  1     1 -1 -1 -1
 1  1  1  1  1 -1 -1 -1    -1  1 -1 -1
 1  1  1  1  1  1  1  1    -1 -1  1 -1
 1 -1 -1 -1 -1 -1  1 -1     1 -1 -1 -1
 1 -1  1  1  1  1  1 -1     1 -1 -1 -1
 1  1  1  1  1  1  1 -1    -1  1  1 -1
 1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1 -1  1  1  1  1  1  1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1  1
-1 -1 -1 -1 -1  1  1  1    -1  1 -1  1
-1  1  1 -1 -1  1  1  1     1 -1 -1 -1
 1 -1  1 -1  1 -1 -1 -1    -1  1 -1  1
-1 -1 -1  1 -1  1  1  1     1 -1 -1 -1
-1  1  1  1  1  1  1  1    -1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1  1 -1 -1
-1  1 -1 -1 -1 -1  1 -1    -1 -1  1  1
-1  1  1  1 -1  1  1 -1     1 -1 -1  1
-1  1 -1  1 -1  1 -1  1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1 -1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1    -1 -1  1 -1
-1  1  1 -1 -1  1  1 -1     1 -1 -1  1
-1  1  1 -1 -1  1  1 -1     1 -1 -1 -1
-1 -1  1  1  1  1  1  1    -1 -1  1 -1
-1  1 -1 -1 -1 -1 -1 -1    -1  1 -1 -1
-1  1 -1 -1 -1 -1 -1 -1     1 -1 -1  1
-1 -1 -1  1 -1  1  1 -1    -1 -1 -1 -1
 1  1  1 -1 -1  1  1 -1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1     1 -1 -1 -1
 1 -1 -1  1  1 -1 -1 -1     1  1 -1 -1
-1  1 -1 -1 -1 -1 -1  1    -1  1 -1 -1
-1 -1 -1  1  1 -1 -1  1    -1 -1 -1  1
 1 -1 -1  1  1 -1 -1  1    -1 -1  1 -1
 1  1  1  1  1  1  1  1    -1  1 -1 -1
 1  1 -1 -1 -1 -1 -1  1     1 -1 -1  1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
-1  1  1  1  1  1 -1  1    -1 -1 -1  1
 1  1  1  1  1  1 -1  1    -1  1 -1  1
 1 -1 -1  1 -1  1 -1  1    -1  1 -1 -1
-1  1 -1  1 -1  1  1  1    -1 -1  1 -1
-1  1 -1 -1 -1 -1  1 -1     1  1  1 -1
-1 -1 -1 -1 -1 -1 -1  1     1 -1 -1  1
 1 -1  1  1 -1  1  1 -1    -1  1 -1 -1
 1  1  1 -1  1 -1 -1 -1     1 -1  1 -1
 1 -1  1  1  1  1  1  1    -1 -1 -1 -1
 1 -1  1 -1 -1  1  1  1     1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1     1 -1  1 -1
-1 -1 -1  1 -1  1 -1  1     1 -1 -1 -1
 1  1  1 -1 -1  1  1 -1    -1 -1 -1  1
-1  1  1 -1 -1  1 -1  1     1 -1 -1 -1
-1  1  1 -1  1 -1 -1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1  1  1 -1  1 -1 -1  1    -1 -1  1  1
-1  1  1 -1 -1  1  1 -1     1 -1 -1 -1
-1 -1 -1  1 -1  1  1  1     1 -1  1 -1
-1  1 -1 -1 -1 -1 -1 -1    -1 -1  1 -1
-1  1  1 -1 -1  1  1  1    -1  1 -1 -1
-1 -1 -1  1 -1  1  1 -1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1 -1 -1     1 -1 -1 -1
-1  1  1 -1 -1  1  1  1    -1  1 -1 -1
 1 -1  1 -1 -1  1  1  1    -1 -1 -1  1
 1 -1  1 -1  1 -1 -1 -1    -1 -1 -1 -1
```

-1 1 1 1 1 1 -1 1	-1 -1 -1 -1
1 1 1 -1 1 1 1 -1	-1 -1 -1 -1
-1 1 1 1 1 1 1 -1	1 -1 1 -1
1 1 1 1 1 1 -1 -1	-1 -1 -1 1
1 1 1 -1 -1 1 1 -1	1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
1 1 1 1 1 1 1 -1	-1 -1 -1 -1
1 -1 -1 -1 -1 -1 -1 1	1 -1 1 -1
-1 -1 -1 1 1 -1 -1 1	-1 -1 1 1
1 -1 -1 -1 -1 -1 1 -1	-1 -1 -1 -1
-1 1 1 -1 -1 1 1 1	-1 1 -1 -1
1 -1 1 -1 -1 1 1 1	-1 -1 -1 1
-1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
-1 -1 -1 1 1 -1 -1 -1	-1 -1 -1 -1
1 -1 1 1 1 1 1 1	-1 1 -1 -1
1 1 1 -1 -1 -1 -1 -1	-1 -1 1 -1
-1 1 1 -1 1 -1 1 -1	1 1 -1 -1
1 -1 1 -1 1 1 1 -1	-1 -1 -1 1
1 -1 1 -1 1 -1 -1 -1	-1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 1
-1 -1 1 1 1 1 1 -1	1 -1 -1 -1
1 -1 -1 1 -1 1 1 1	1 1 1 -1
-1 -1 -1 -1 -1 -1 1 -1	-1 1 1 1
-1 -1 -1 -1 -1 -1 -1 -1	-1 1 -1 1
-1 1 1 -1 -1 1 1 1	-1 -1 -1 1
-1 1 -1 1 -1 1 1 1	1 -1 -1 -1
1 1 1 1 1 1 1 1	1 -1 1 1
-1 1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
-1 -1 -1 1 1 -1 -1 1	1 -1 -1 -1
1 1 1 -1 -1 -1 1 -1	1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
-1 1 1 -1 -1 1 1 -1	-1 -1 -1 -1
-1 1 1 -1 1 -1 1 -1	-1 1 -1 -1
1 -1 -1 -1 -1 -1 -1 -1	-1 1 -1 1
-1 -1 1 1 1 1 1 1	-1 -1 1 -1
1 -1 -1 1 1 -1 -1 1	-1 -1 1 -1
1 -1 -1 1 1 -1 -1 1	1 -1 1 -1
1 1 -1 1 -1 -1 -1 -1	-1 1 -1 -1
1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
-1 1 1 1 1 1 -1 1	-1 -1 -1 1
1 1 1 -1 1 -1 -1 -1	-1 1 -1 -1
-1 -1 -1 1 -1 1 1 -1	-1 -1 -1 -1
1 1 1 -1 -1 1 -1 1	1 -1 -1 1
1 1 1 1 1 1 1 1	-1 -1 -1 -1
1 -1 1 1 1 1 1 1	-1 1 -1 -1
1 1 1 -1 1 -1 -1 -1	-1 1 -1 -1
1 -1 1 1 1 1 -1 1	-1 1 -1 1
-1 1 -1 -1 -1 -1 -1 1	-1 1 -1 1
1 -1 1 1 1 1 1 1	-1 -1 -1 1
-1 1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
-1 -1 -1 1 -1 -1 -1 -1	-1 -1 -1 1
1 -1 -1 -1 -1 -1 -1 -1	1 -1 -1 1
-1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1
1 -1 1 -1 -1 1 -1 1	1 -1 -1 1
-1 -1 -1 -1 -1 -1 -1 1	-1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 1	-1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 1	-1 1 -1 1
-1 -1 -1 -1 -1 -1 1 -1	1 1 -1 -1
-1 1 1 1 1 1 1 1	-1 -1 -1 -1
1 -1 1 1 -1 1 1 -1	-1 -1 -1 -1
1 1 1 1 1 1 -1 1	-1 -1 1 1
1 -1 -1 -1 -1 -1 -1 -1	1 -1 1 -1
-1 -1 -1 1 1 -1 -1 -1	1 -1 -1 -1

```
 1  1  1  1  1  1  1 -1    -1  1 -1 -1
 1  1  1 -1  1 -1 -1  1     1 -1  1  1
-1 -1 -1 -1 -1 -1 -1 -1    -1  1 -1  1
-1 -1 -1 -1 -1 -1  1 -1    -1  1 -1 -1
-1  1 -1  1 -1  1  1 -1    -1  1  1 -1
 1 -1 -1 -1 -1 -1 -1  1     1 -1 -1 -1
 1 -1  1 -1  1 -1 -1 -1    -1 -1 -1 -1
-1 -1  1  1 -1  1  1  1    -1 -1 -1 -1
 1  1  1  1  1  1  1  1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1  1 -1     1 -1 -1  1
 1 -1  1 -1  1 -1 -1 -1    -1 -1  1 -1
-1 -1 -1  1 -1  1  1 -1     1  1 -1  1
 1 -1  1  1  1  1  1 -1    -1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1 -1 -1 -1
-1  1 -1  1  1 -1  1 -1    -1 -1  1 -1
 1  1  1 -1  1 -1 -1 -1    -1 -1 -1 -1
-1  1  1 -1  1 -1  1 -1    -1 -1 -1  1
-1 -1  1 -1 -1  1  1 -1    -1 -1 -1 -1
-1  1  1 -1  1 -1 -1  1    -1 -1 -1  1
 1 -1  1 -1 -1 -1  1  1     1 -1 -1  1
-1 -1 -1 -1 -1 -1 -1  1    -1  1  1 -1
-1  1  1 -1 -1  1 -1 -1    -1  1 -1 -1
-1 -1  1  1  1  1  1  1     1 -1 -1  1
 1 -1 -1  1  1 -1 -1  1     1 -1 -1 -1
 1 -1 -1 -1  1 -1 -1  1     1  1 -1  1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1  1
-1 -1 -1 -1 -1 -1  1 -1    -1 -1 -1 -1
-1 -1  1 -1 -1  1 -1  1    -1 -1  1 -1
-1  1 -1  1 -1  1  1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1  1 -1 -1 -1 -1 -1  1    -1 -1  1  1
 1 -1  1  1  1  1  1  1     1 -1  1  1
 1 -1 -1  1  1  1 -1  1    -1 -1  1 -1
 1  1  1  1  1  1  1 -1    -1 -1 -1 -1
 1  1  1  1  1  1 -1  1    -1 -1  1 -1
 1  1  1 -1  1 -1 -1 -1     1 -1 -1  1
-1  1  1 -1 -1 -1  1 -1    -1  1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1    -1 -1  1 -1
-1  1 -1 -1 -1 -1 -1  1     1 -1 -1 -1
-1  1  1  1  1  1 -1  1     1 -1 -1 -1
 1 -1  1 -1  1 -1  1 -1    -1 -1 -1 -1
-1 -1 -1  1 -1  1  1  1    -1  1 -1 -1
 1  1  1 -1 -1  1  1  1    -1  1 -1  1
-1 -1 -1  1 -1  1 -1  1     1 -1 -1 -1
 1  1  1 -1 -1  1 -1  1     1 -1 -1 -1
-1  1 -1  1  1 -1 -1  1    -1  1 -1 -1
 1  1  1  1  1  1  1  1    -1 -1 -1 -1
-1  1 -1 -1 -1 -1 -1  1    -1  1 -1  1
-1  1  1  1  1  1  1 -1    -1 -1 -1  1
-1 -1 -1 -1 -1 -1 -1  1     1 -1  1 -1
-1  1 -1 -1 -1 -1  1 -1    -1  1  1 -1
-1 -1 -1 -1 -1 -1 -1  1    -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1     1  1 -1 -1
-1 -1  1 -1  1 -1  1 -1    -1 -1 -1 -1
-1  1  1 -1  1 -1 -1  1     1 -1 -1  1
-1 -1 -1  1 -1  1  1  1    -1  1  1 -1
 1 -1 -1 -1 -1  1  1  1     1 -1 -1 -1
 1  1  1  1  1  1 -1  1     1  1  1  1
 1  1  1 -1 -1  1  1  1     1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1 -1    -1 -1 -1 -1
-1  1  1  1  1  1 -1 -1     1  1 -1  1
-1  1 -1  1  1 -1 -1  1    -1 -1 -1 -1
 1 -1 -1 -1 -1 -1  1 -1    -1 -1 -1  1
```

MONTANA STATE UNIVERSITY LIBRARIES
3 1762 10116165 9

HOUCHEN
BINDERY LTD
UTICA/OMAHA
NE.