Cambridge Hydrodynamics Report 643

DEVELOPMENT OF A NEW TECHNIQUE FOR IMAGE RECONSTRUCTION, ENHANCEMENT, AND VISUALIZATION

— Final Report on AFOSR Contract F49620-90-C-0028 —

Steven A. Orszag, Principal Investigator

February 1993

Cambridge Hydrodynamics, Inc.
P. O. Box 1403
Princeton, NJ 08542
(609) 683-1515

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
Development of a New Technique for Image Reconstruction, Enhancement, and Visualization (U)

5. TYPE OF REPORT & PERIOD COVERED
FINAL 15APR90-14OCT92

6. PERFORMING ORG. REPORT NUMBER
CHI Report 643

7. AUTHOR(s)
Steven A. Orszag, Principal Investigator

8. CONTRACT OR GRANT NUMBER(s)
F49620-90-C-0028

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Cambridge Hydrodynamics, Inc.
P. O. Box 1403
Princeton, NJ 08542

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
AFOSR-TR- 93 0268

11. CONTROLLING OFFICE NAME AND ADDRESS
Air Force Office of Scientific Research
110 Duncan Ave, Suite B115
Bolling AFB, DC 20332-0001

12. REPORT DATE
February 1993

13. NUMBER OF PAGES
150

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*
SAME AS 11

15. SECURITY CLASS. *(of this report)*
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*
Unclassified — Unlimited
APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

Original contains color plates: All DTIC reproductions will be in black and white.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*
SAR

18. SUPPLEMENTARY NOTES
93 4 20 101

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
Triangulation Image reconstruction Image enhancement Edge enhancement
Surface mapping Delaunay triangulation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
The problem of describing and reconstructing a surface, both analytically and visually, has been extensively addressed in this work. Various algorithms have been developed, tested, and compared and several conclusions have been reached. They are: (1) if the surface information is given on a uniform grid, the surface is best described by a set of contour plotting routines that exploit the GL graphics libraries both in a solid as well as a wire-mesh framework; (2) if the surface information is given on a non-uniform grid, the surface can be described by Delaunay triangulation, a sophisticated graphics computer, or by interpolation back to a uniform grid. Numerous examples are given which show the advantages of these methods for specific applications.

DD FORM 1 JAN 73 1473

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

DEVELOPMENT OF A NEW TECHNIQUE FOR IMAGE RECONSTRUCTION, ENHANCEMENT, AND VISUALIZATION

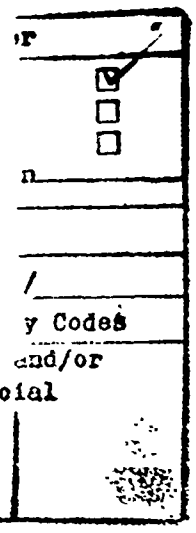
Principal Investigator: Dr. Steven A. Orszag

1. INTRODUCTION

The problem of describing and reconstructing a surface, both analytically and visually, has been extensively addressed in this work. Various algorithms have been developed, tested, and compared and several conclusions have been reached. They are: (1) If the surface information is given on a uniform grid, the surface is best described by a set of contour plotting routines that exploit the GL graphics libraries both in a solid as well as a wire-mesh framework; (2) If the surface information is given on a non-uniform grid, the surface can be described by Delaunay triangulation, a sophisticated graphics computer, or by interpolation back to a uniform grid. Each of these methods possesses certain advantages over the others. In the event of insufficient information on the surface, two basic methods have been employed: (1) an algorithm to fill in "missing" information that employs Delaunay triangulation; and (2) interpolation amongst various grid points of known data assuming that the surface is relatively smooth. In the case of distinct missing information on outliers (regions of the surface possessing extremal properties missing from the original surface), no progress has been made.

2. TECHNICAL BACKGROUND

In general, a surface is described by a function, such as f(x,y,z)=const Alternatively, a surface can be described by z = g(x,y), with the understanding that g(x,y) can take on multiple values of the surface folds back on itself. A typical area of interest to the Air Force is a surface reconstruction of photographs taken from high altitude. In some portions of the photographs where high resolution is needed, certain blurriness associated with absence of resolution may need to be improved. Furthermore,



the need to remove some white noise from the picture may result in a distinct sharpening of the picture. Statistically, as long as the surfaces involved are very smooth, standard regression analysis could very successfully yield the desired results. However, the information that one wants to obtain are more often than not associated with some characteristics that are absent from the general smooth background of the images.

There are two distinct problems in surface reconstruction:

(i) Decomposition of the surface into elements.

(ii) Smoothly connecting these elements by interpolation.

The best solution to these problems is different for uniform grids than it is for non-uniform grids.

For non-uniform grids, the first problem can be solved by the Voronoi-Delaunay construction. Each vertex is surrounded by a Voronoi cell, defined as that cell that contains points which are closer to this vertex than to any other. The boundaries between neighboring Voronoi cells are lines, equidistant from a pair of vertices. Connecting all such pairs by links generates a Delaunay triangulation. We exploit this geometrical construction by observing that each triangle (abc) involves vertices which are Voronoi neighbors. There is a point Y where all three Voronoi cells (a,b,c) touch. By construction, all three vertices a,b,c are equidistant from Y so that these points are on the circle centered on Y. No other vertex can exist inside this circle by definition of the Voronoi cells. This property has been used in the construction of an incremental algorithm for Delaunay triangulation. We have developed an algorithm that accomplishes this triangulation recursively. Strictly speaking, the task is to add a point to an existing triangulation of N points. An added point exists and is contained inside one of the existing triangles. If we connect the new point to the vertices of the triangle in which it lies, we get the zeroth approximation of the triangulation of N+1 points. This triangulation is good away from the new point, but must be rebuilt in the local neighborhood of the new point. The maximum area of reconstruction is given by the

circle centered on Y, and in general, it requires only a finite number of bond flips (triangle reconstructions). A recursive flipping algorithm can be derived and is very fast. The most costly operation in this incremental Delaunay triangulation is the search for the triangle surrounding the new point. Straightforward scanning takes O(N) time, but preprocessing the data accelerates the search by the following strategy.

One can order the vertices along any specific curve so that neighbors on the curve will be always close to each other on the plane. (This is a version of the traveling salesmen problem.) Such an ordering of initial points takes O(N log N) time. This ordering is needed to be performed only once, after which various progressive surfaces can be drawn using the same coordinate system. Degenerate cases, when the points are too close to each other or the (N+1)*st* point is too close to the side of the N*th* triangulation, are treated separately to prevent numerical instabilities.

The second approach is based on the graphics implemented in the graphics terminals of Evans and Sutherland. The points are loaded on the system and are displayed simultaneously with x,y, z coordinates all belonging to the same surface. The surface can be rotated around each of the three axes and cutting planes have been introduced to isolate specific projections of the surface on a plane of choice. On a plane of choice, curves have been interpolated using optimal bicubic splines. The points are added on the splines and the files of the individualized planes get re-assembled. A perpendicular plane to the original set of parallel planes is being introduced and the added points in each of the previous planes is compared to the initial set of points on this cut plane. A least-squares estimator is performed to decide on admissibility or inadmissibility of the added points. If the set of points added to the global grid is admissible, a point-by-point elimination process for the added points is invoked until the set of added points becomes admissible. The set of points that are then admissible per

perpendicular plane are then added to the global grid and the process may be repeated until no more points are admissible. At that stage, the entire file is re-loaded on the graphics system and a surface reappears. The surface can be described by the actual collection of dots, wire-mesh, or by standard graphics shading routines on the graphics processors. We are now in the process of adapting these algorithms to the SGI GL libraries.

The third case for the non-uniform grid can be handled in two different ways. First, a uniform grid is imposed in the x-y plane. At each grid point, a characteristic radius is introduced. Each point in the x-y plane, x_i, y_i is taken into account. The number of points p in the enclosed domain is recorded and a pth Lagrange interpolation scheme is introduced to compute the value z associate with the said grid point. When all the uniform grid point values have been obtained, the interpolation scheme is checked against each existing original point. In other words, each original point is attempted to be recomputed to check for accuracy. This is done with variable orders in order for outliers estimates to be rejected. In the second case, at any vertex in the Delaunay triangulation, an arbitrary number of triangles can meet and the property of smoothness at the vertex requires a large number of local parameters. We have the following construction that satisfies all the required smoothness conditions. The Gauss formula for planes of the corresponding triangles meeting at a vertex has been used. The smooth function so obtained passes through all its neighbors. This function is usually a sum of about six terms. Three of such functions are connected inside each triangle by a similar Gauss interpolation resulting in an explicit local set of rational functions within each triangle matching their neighbors with any prescribed smoothness. The addition of the interpolated points to the original triangulated surface is composed by the set of points at the center of the Delaunay circles, i.e. vertices of the Voronoi graph which are always located in the voids of the Delaunay triangulation. Adding these circle centers does not

require a search and is numerically stable. It roughly doubles the number of points omitting the points whose Delaunay circle radius is smaller than a given criterion to avoid condensation of points. After a few steps, the plane is completely covered and the process is complete.

The issue of describing a surface on a uniform grid is different in nature. An aerial photography or topographic description of a surface can in principle be described on a uniform grid since the data gathering can be digitized according to the data gatherer. At this juncture, a missing observation on the uniform grid is interpolated using standard Lagrange interpolations and data contained within noise can be filtered out by invoking standard FFT techniques. In particular, a combination of low and high-pass filters can isolate specific structures by sharpening the domain of resolution. It is believed that this technique may have the potential of image enhancement.

One of the major goals in this study is the ability to characterize an irregular surface, in particular, non-uniform surfaces obtained either from exterior sets of observations or from solutions of partial differential equations of moving fronts or equipotential surfaces. For example, the system of detecting irregular conductivity of an internal human tissue is dependent on surface potential and conductivity. A solution of the potential problem inside the bulk with isolation of equipotential surfaces is hoped to improve visualization of internal irregular tissues. It is for such surfaces that the non-uniform grid methodologies developed here will be useful.

3. IMPLEMENTATION

We have developed the algorithms specified in Tasks 1-3 of the original proposal. These are exemplified by the attached source code in Appendix I. For Task 4, we have developed a menu-driven user-based interface that is displayed with all its functions in

the attached plots. The use of this interface and code is elementary and requires minimal learning time. The code interface is implemented by use of a mouse within the graphics window. For Task 5, we provide various sets of examples of non-uniform as well as uniform grid systems. The uniform grid graphics visualization displays a system borrowed from microelectronics of contact hole printing when the surface of the electromagnetic energy intensity is displayed above a given surface. Multi-resolution is used in these calculations to address the requirements of circuit designers. Furthermore, we attach two different circuit designs (see Figs. 1-2) obtained for uniform grid implementation of two separate logic patterns for wiring and memory triggers. In both cases, GEOFORM gave a reasonably satisfactory description of the circuit and its associated resolution.

For the non-uniform grid systems, we have tested this method on geologic data, Martian topography, vortex sheets in hydrodynamics and also on a sculpture reconstructed from a few hundred points. Sometimes smoothing was needed, i.e., the points were iteratively moved towards the local second-order surfaces, least-square fitted to the nearest neighbors. Since this fit was local, it did not erase the small details at the surface of larger structures. Everything was smoothed at its own scale. This algorithm is surprisingly efficient. The Mars topography reconstruction required only 2% of randomly chosen data points. The sculpture was reconstructed from just 600 points, and the roll-up of a vortex sheet was simulated using only 1000 points. In Figure 3, we display the Delaunay triangulation of the initial mask. The reconstructed surface is displayed next to this triangulation and the internal variations in the density of the triangulation manifests in different shading to create the necessary visualization of the actual original mask. The other two segments of Figure 3 involve the reconstruction of very sharp mountain peaks on the Asian continent where the triangulation density indicated initially a sharp gradient that led to the reconstruction of the Himalayan peaks.

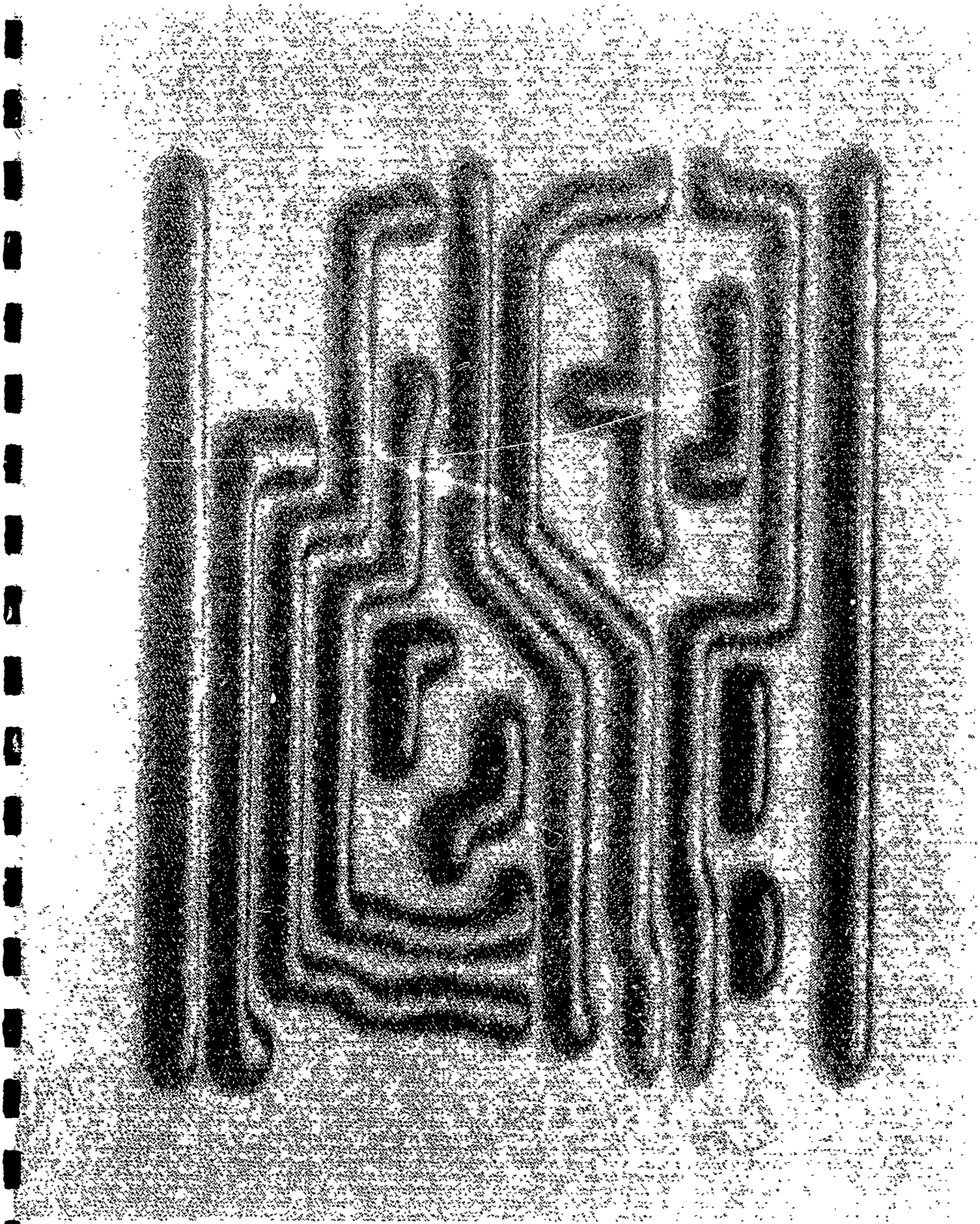
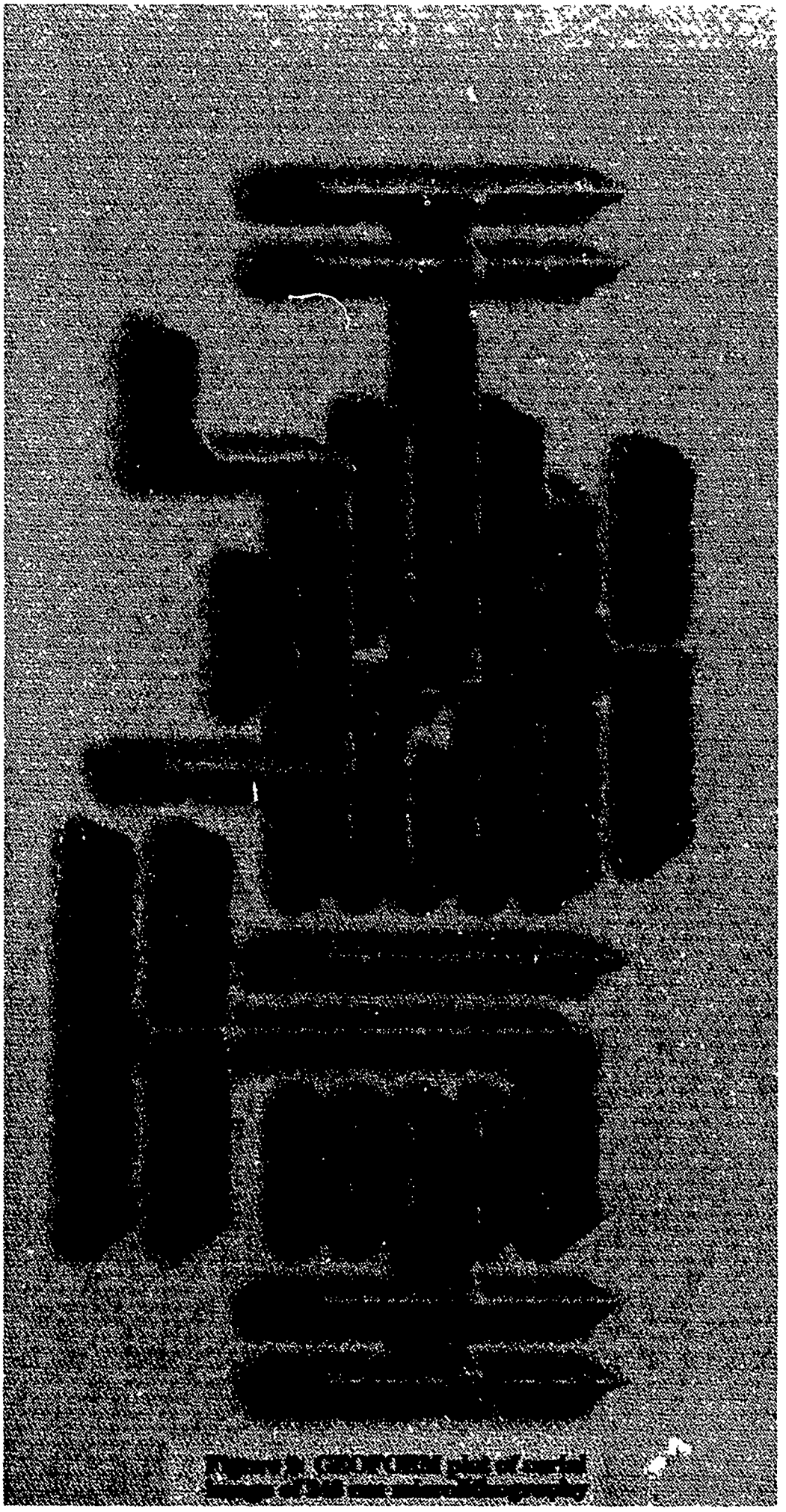


Figure 1: GEOFORM plot of aerial image of 365 nm microlithography



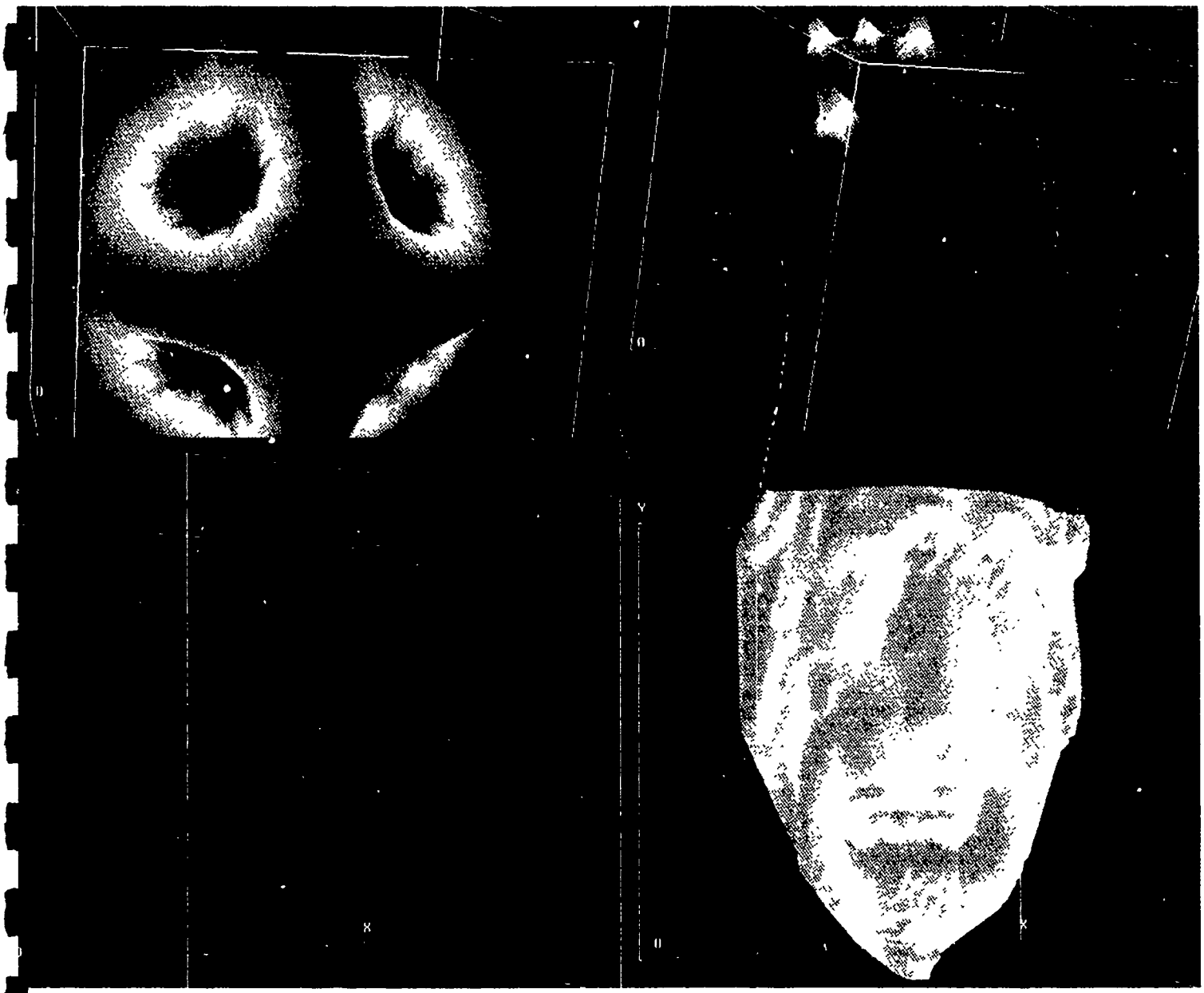


Figure 3: Examples of GEOFORM applications

Higher intensities (higher heights of the mountain peaks) are illustrated in red to emphasize the reconstructed of the missing peak. This technique could be very useful to reconstruct surfaces obtained from high altitude with blurred resolution at the extremes. The fourth quadrant of Figure 3 illustrates a reconstructed sphere with cavities inside. It could be particularly important to use this technique in the simulations of porous media where specific flow channels can be obtained to determine porosity of new materials. The dynamics of pore sizes is under current investigation.

In Figures 4-5, we display graphically the detailed mechanism of edge detection in the GEOFORM code. It is menu controlled and specific points associated with the edge of a given surface are determined locally or globally. In particular, a zoom window provides the ability to zoom in on an edge of interest. The particular example given is that of refining the edge of a 'rabbit' shaped region. It is apparent that even rough resolution suffices to solve the problem. A second graphical option associated with a collection of points is the imposition of the interior Delaunay triangulation embedded in these points.

To explain how we do this, let us consider a collection of points which lie along lines. The points may be given in any order. Our objective is to divide the set of points into subsets, such that all the points in one subset lie along a line. In other words we want to reduce our initial set of points to a set of clusters of close points. We then compress these clusters to "skeletons". These skeletons are represented as graphs with logical links between points. After this compression, one can either compare graphs with a lookup table of standard elements or simply transmit the graph. At each link of the graph one can store the number and mean dispersion of the compressed nearest points, so that one can return to the thick lines after transmission by randomly distributing points alongthe links. As compared to standard methods, this one seems to be especially well suited for graphic objects such as handwriting, fingerprints, space photos or maps.

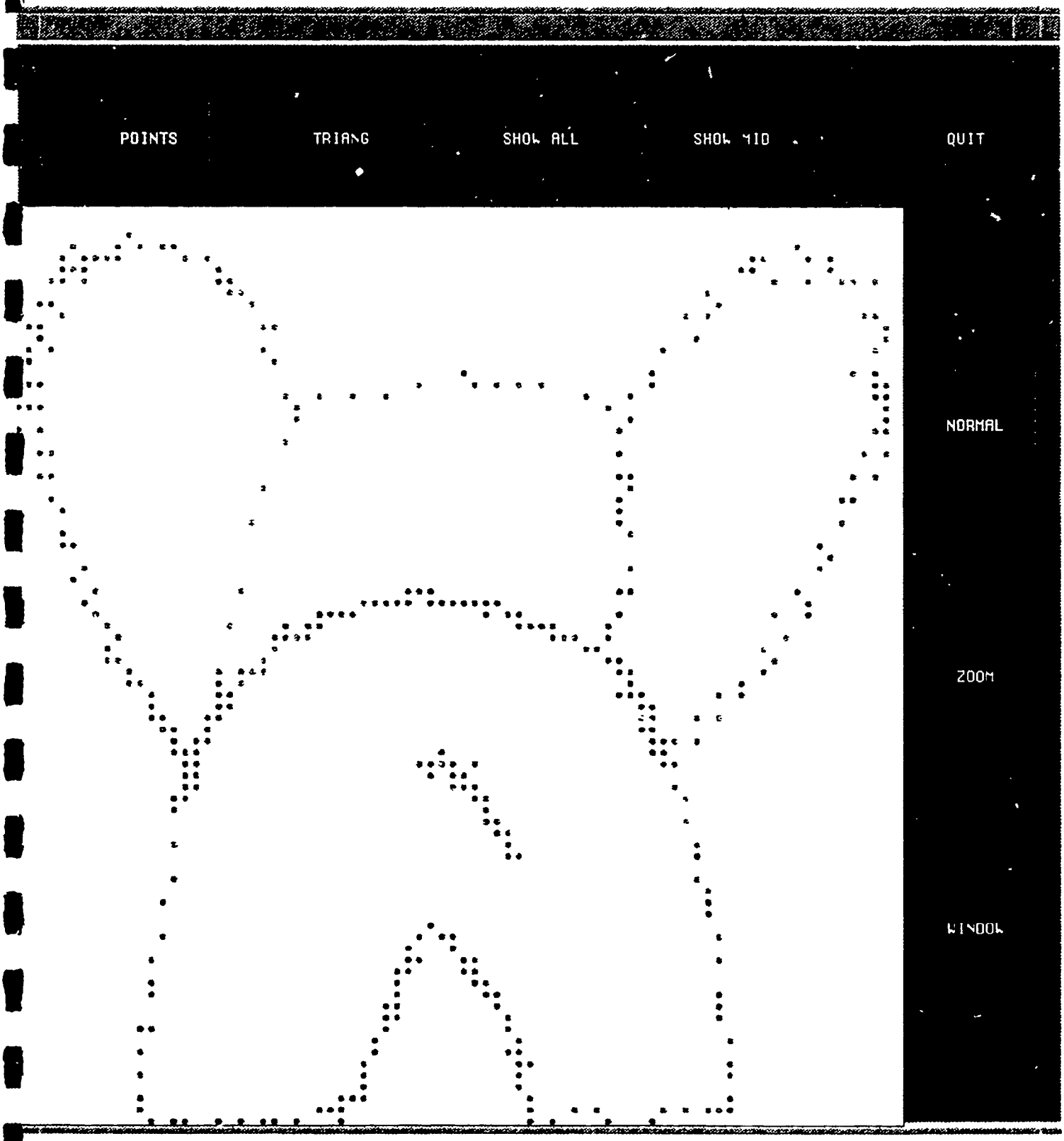


Figure 4: Edge detection: rough boundary of 'rabbit'

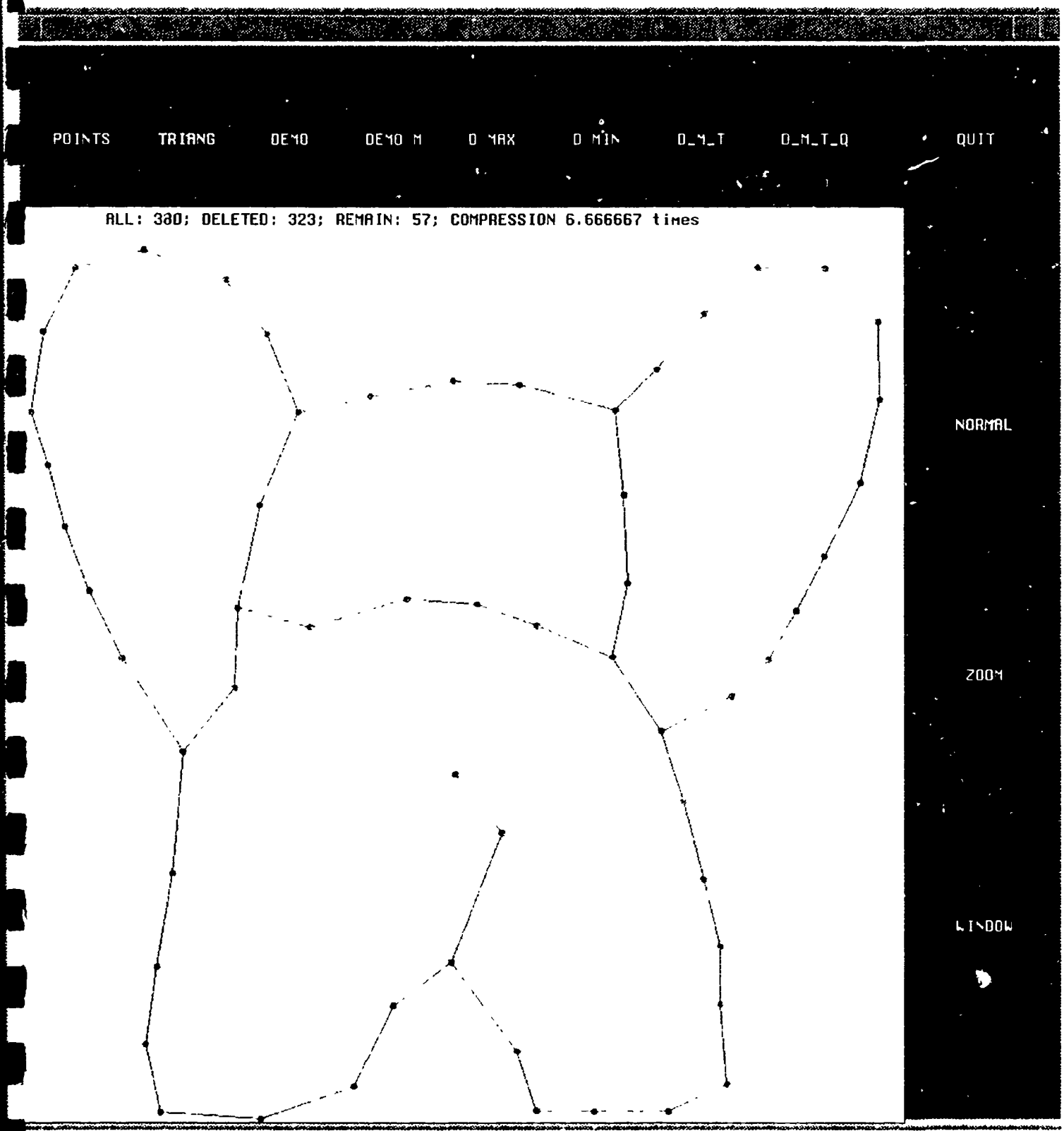


Figure 5: Edge detection: refined boundary of rabbit using GEOFORM

We use the following method. First, we construct the Delaunay triangulation of the set of given points, and then we begin a transformation of the triangulation which will extract the points lying along lines. We have developed a set of functions described below that we apply to our triangulation. The function DELETE MAX EDGE deletes the longest edge of the Delaunay triangulation. The function DELETE MAX EDGE IN TRIANGLE deletes the longest edge, which does not "disconnect" the triangulation, i.e. we delete the edge only if each of its end points has at least three neighbors. The function does nothing if such an edge does not exist. The function DELETE MIN EDGE deletes the shortest edgeof the Delaunay triangulation. The function MERGE MIN EDGE merges the end points of the shortest edge into one point. This new point is placed at the center of mass of the two old points taking into account the number of points which were merged into the points before. The function MERGE MIN TRIANGLE merges the three vertices of the smallest triangle (by area) into one point. The new point is placed at the center of mass of the three vertices, taking into account the number of points which were merged into the vertices before. With these functions in hand, it remains to determine the order of applying them to the triangulation. To explore this question, we developed a tool implemented on IRIS workstations which allows us to choose which function to apply to the traingulation at any time to achieve the best line extraction. The tool collects the information about all steps and prints out the statistics on the working process. This helps to determine the best strategy for applying our functions.

The reconstruction of Martian surfaces is displayed in Figures 6-11. These illustrate extremes obtained from higher Delaunay triangulation densities with missing peaks. A solid surface with proper shading is reconstructed to visualize the actual surface obtained from discrete pieces of information.

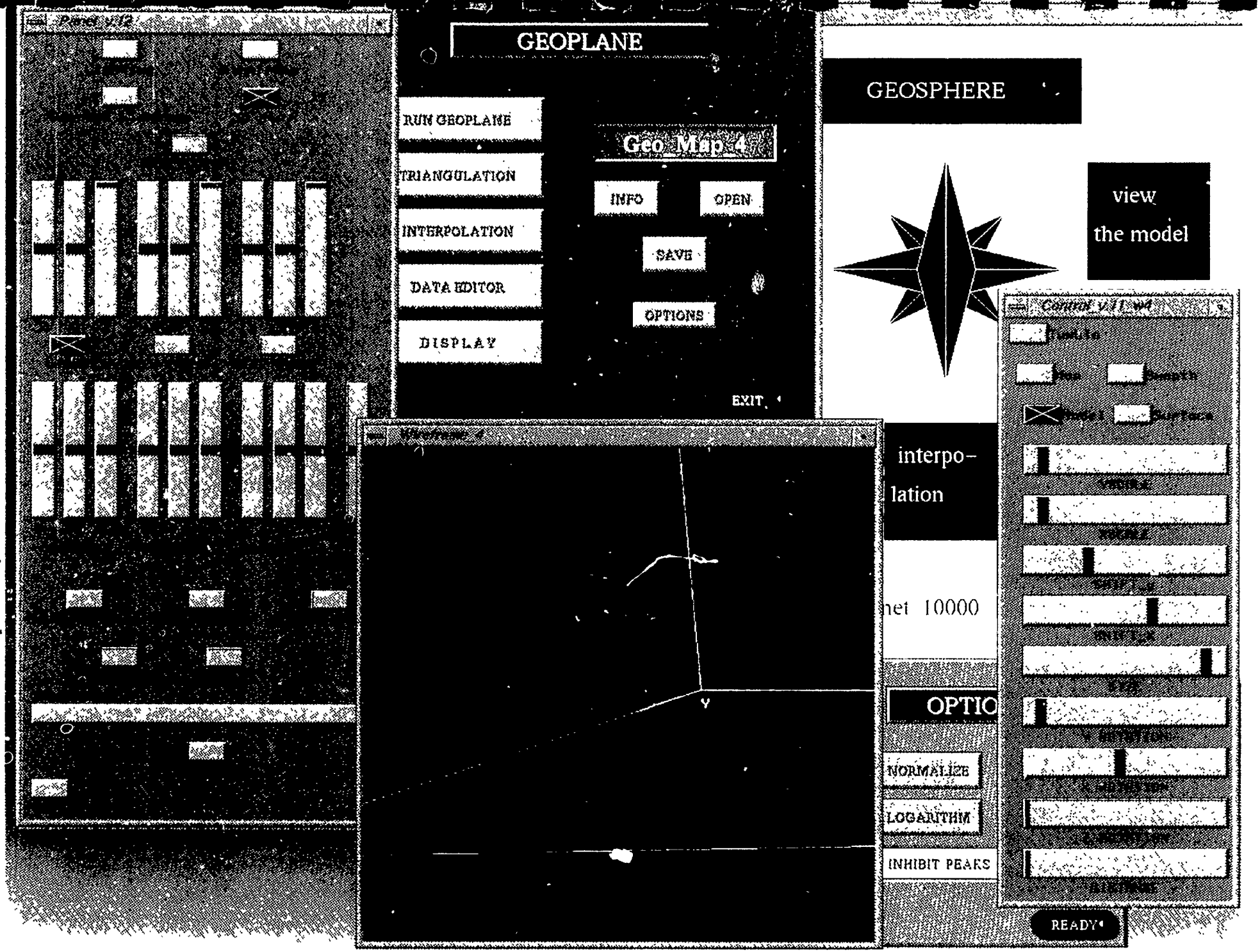


Figure 6: Menu-driven wire-mesh triangulation and refinement

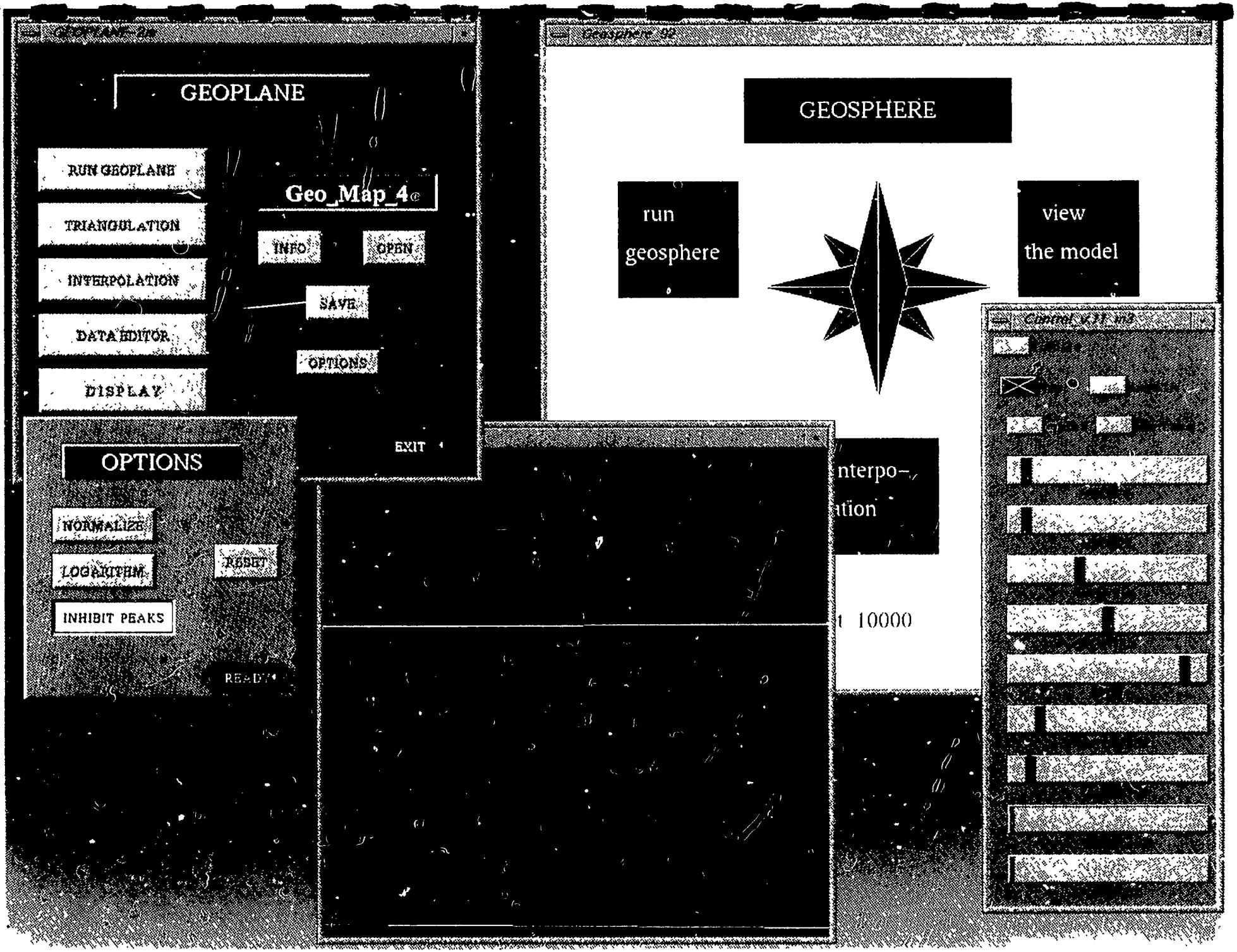


Figure 7: Menu-based control environment of GEOFORM

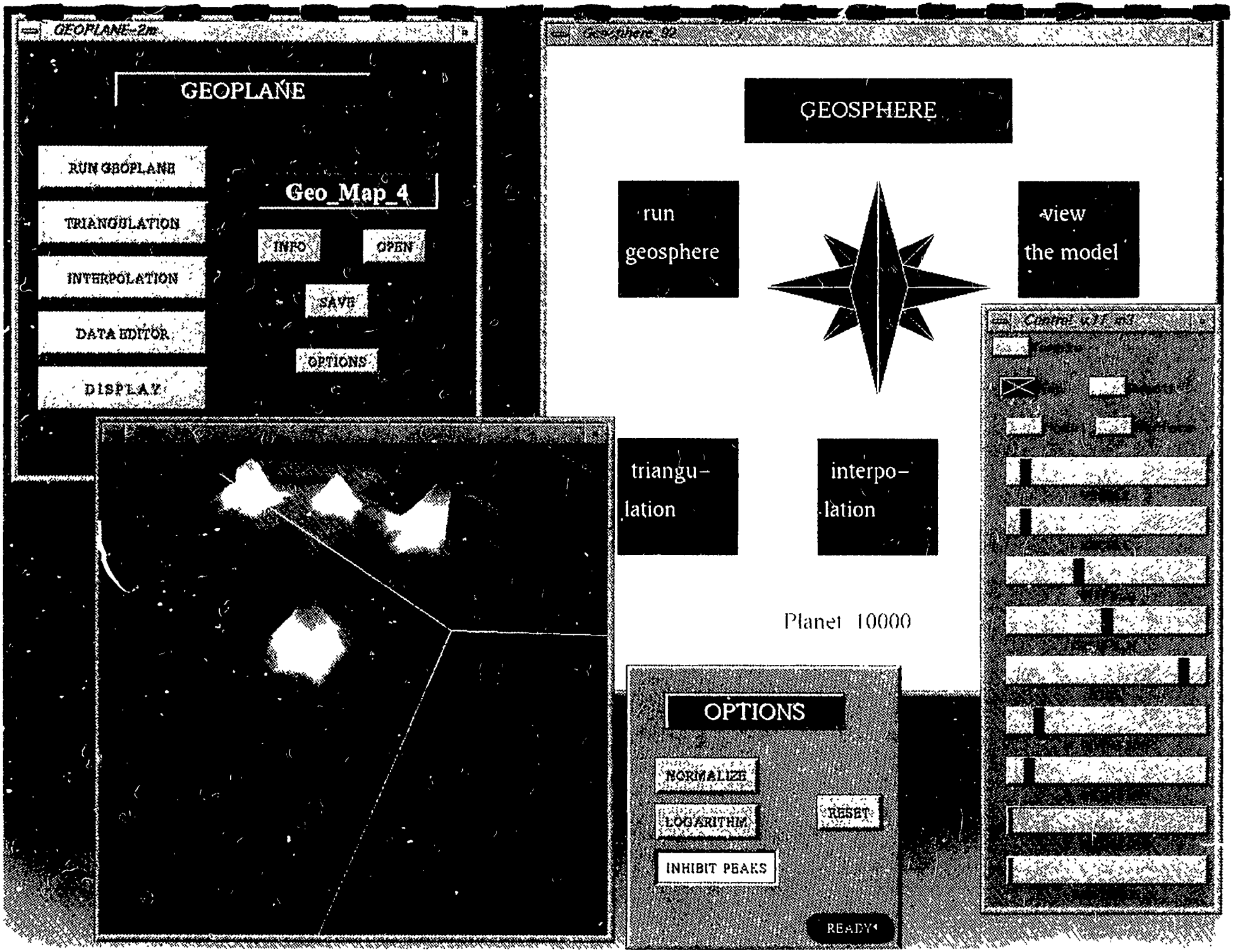


Figure 8: Menu-based topo-mapping environment of GEOFORM

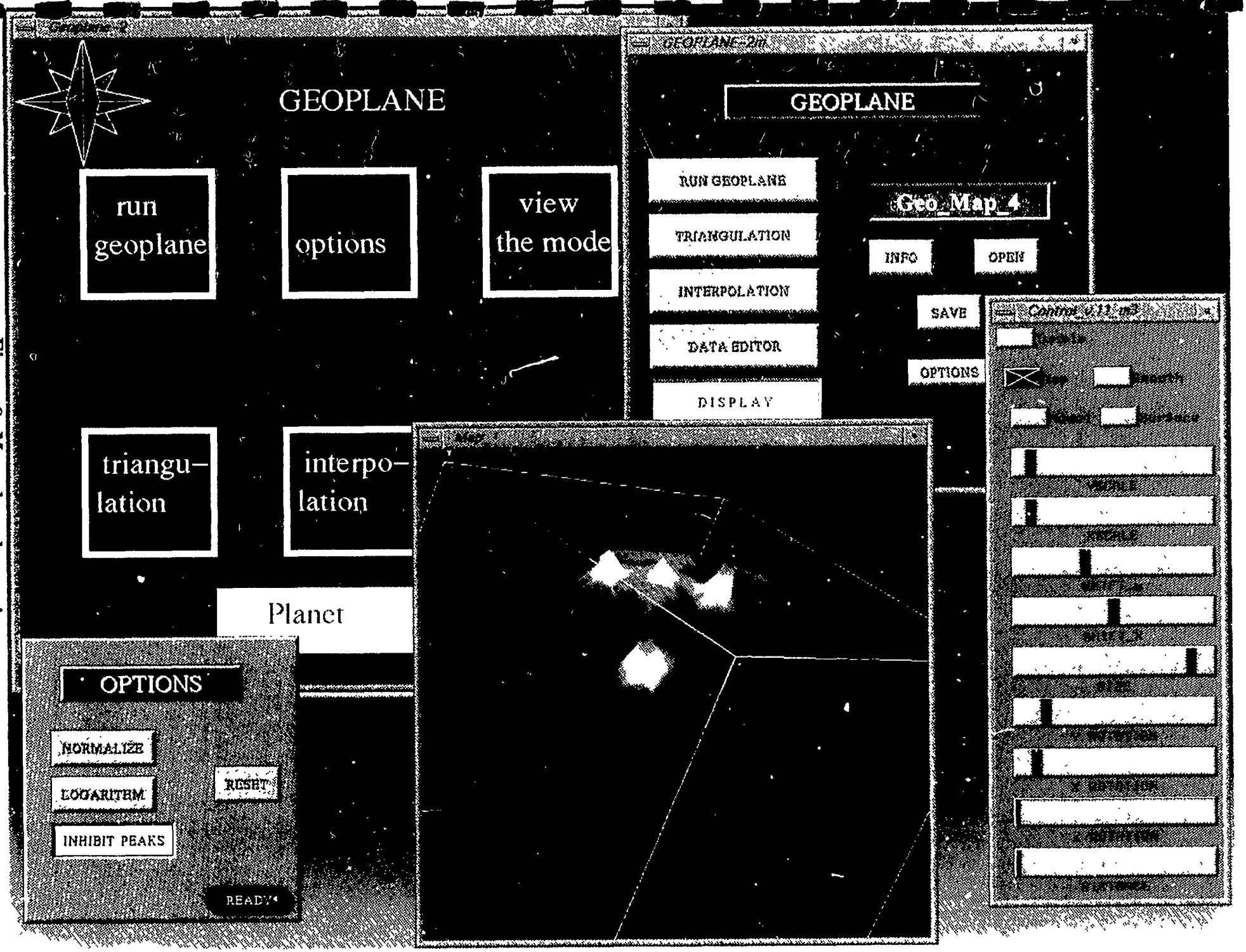


Figure 9: Menu-based mapping environment of GEOFORM

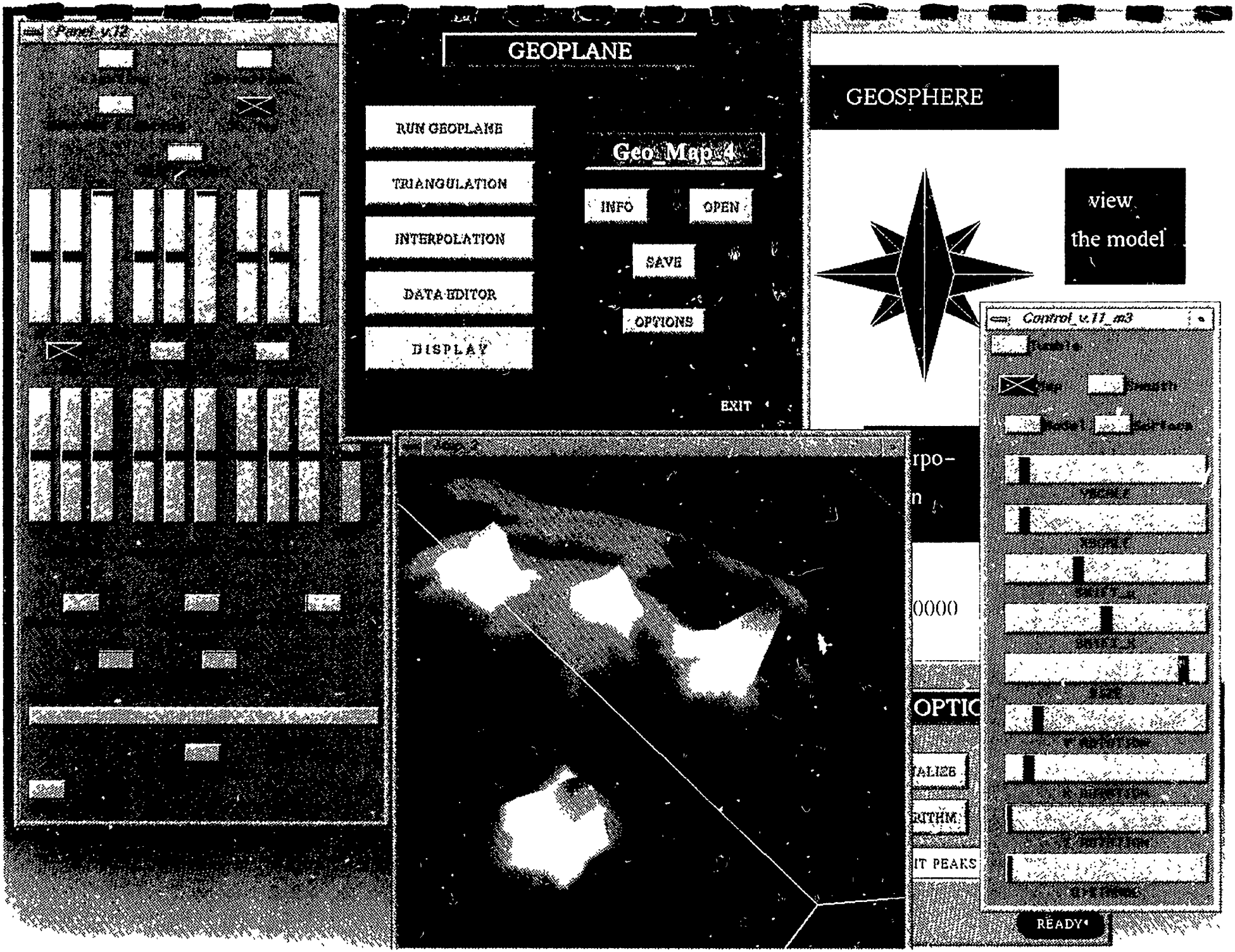
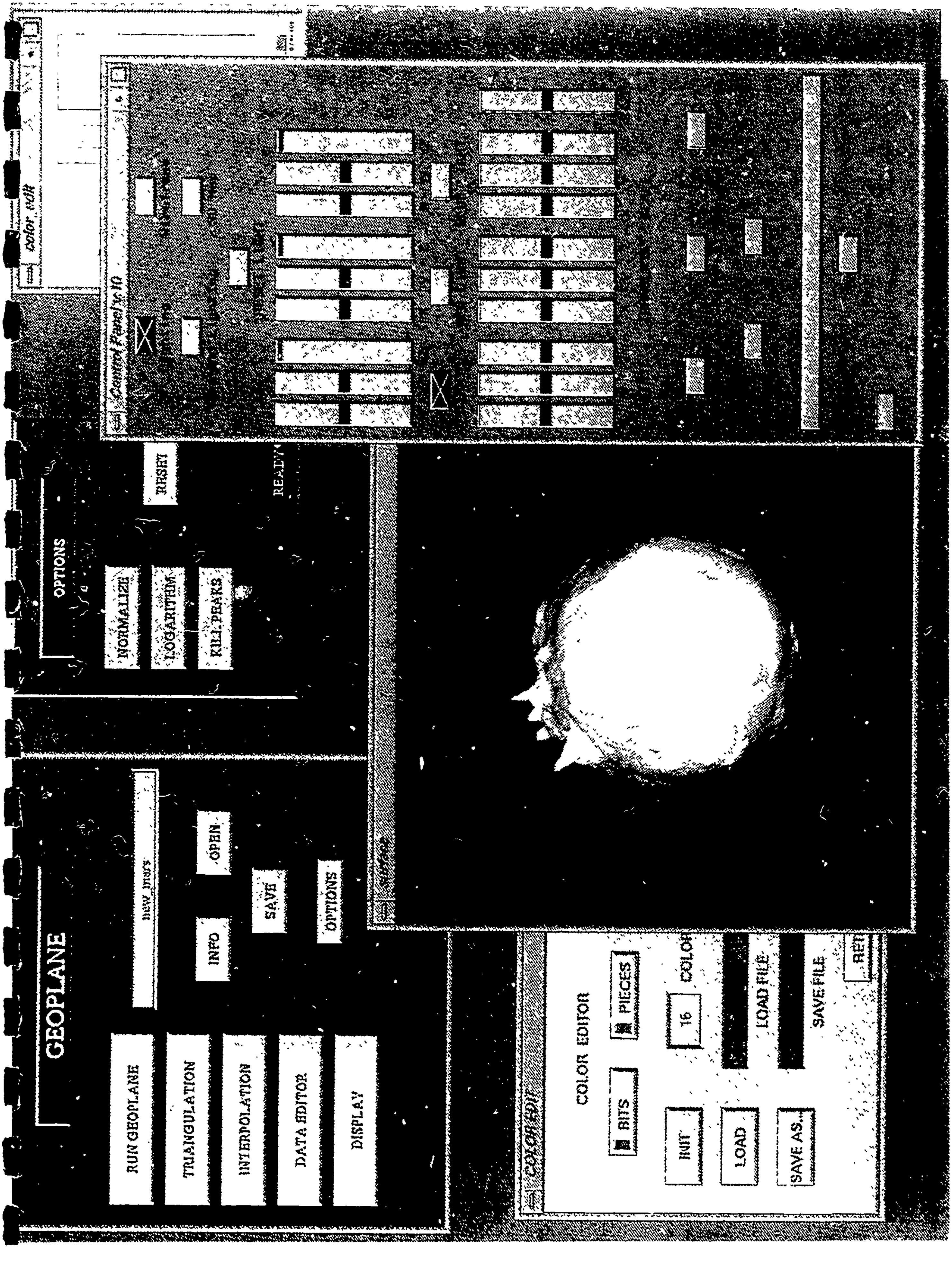


Figure 10: Menu-based zoom-in environment of GEOFORM

Figure 11: Menu-based lighting environment of GEOFORM



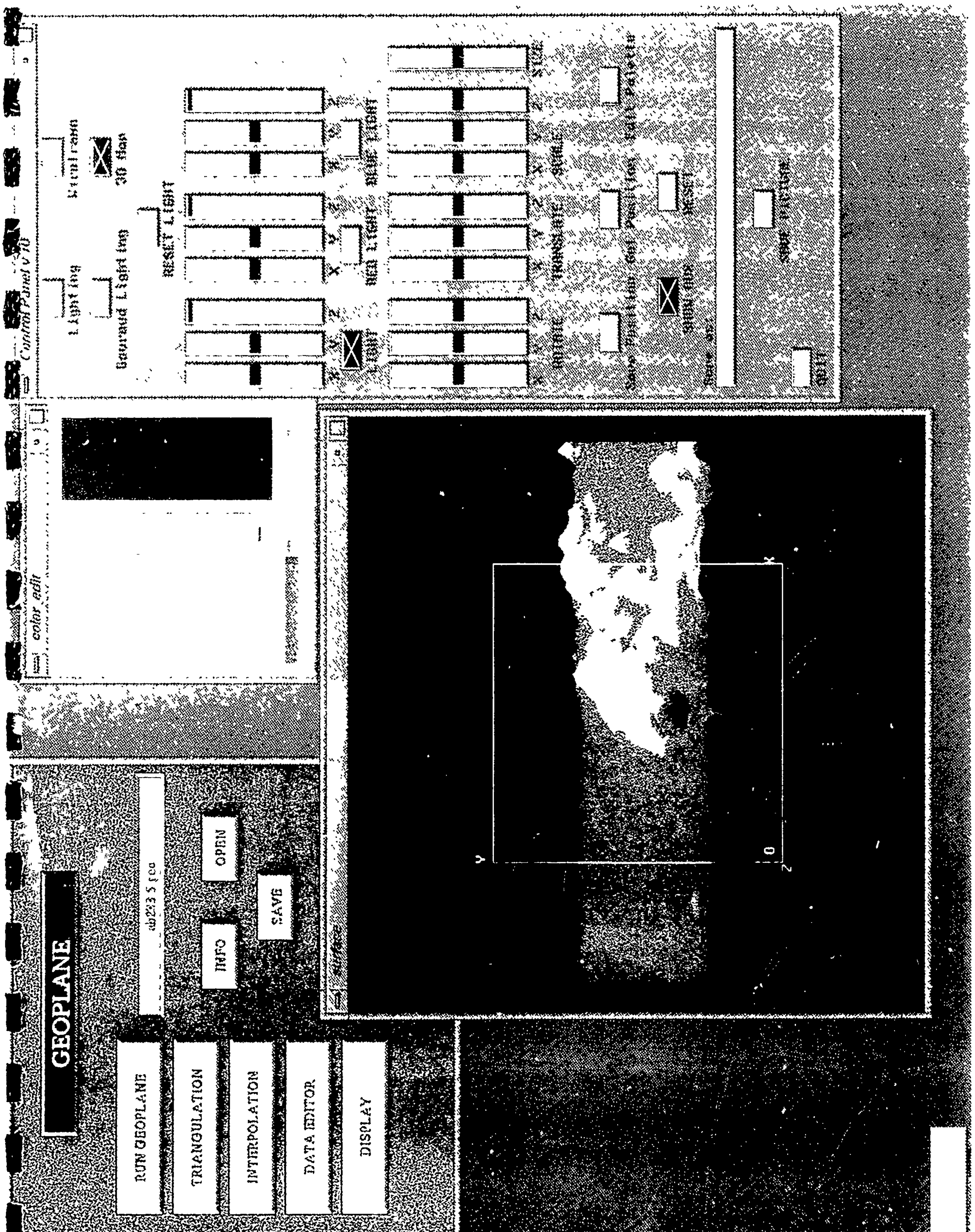
Last, but not least, we describe some of the features associated with the menu-driven graphics system of GEOFORM. In the GEOPLANE panel, we have a button in which the file name to be loaded is inserted. Then, triangulation takes place upon clicking on the triangulation button. After the determination of missing points takes place, the interpolation button gets clicked. Then the data editor is triggered and the display provides the surface. The button labeled 'option' allows either normalization, smoothing by killing peaks, or a logarithmic scale contraction.

Another panel, the control panel, allows changes of light intensities, changes of colors, changes of individual rotation angles, and allows saving of files under the button 'Save Picture'. The color editor selects the bit of the graphics involved and allows the insertion of shading under the control panel instructions. These panels, as well as their sub-panels provide complete graphics control of the surface to be displayed either as a wire mesh or as a shaded solid surface.

The remarkable flexibility of displaying results is best illustrated in the hydrodynamic pictures provided below. In Figures 12-15, we apply GEOFORM to the visualization of flame fronts in turbulent flows. In Figures 16-20, we apply GEOFORM to the visualization of random of multi-frequency Rayleigh-Taylor instabilities occurring at density interfaces.

In conclusion, we have developed new algorithms for data preprocessing and organization, triangulation and interpolation. Numerous tests verify the efficiency and numerical stability of these new algorithms. In particular, the surfaces are glued together from small elements, depending on a finite number of neighboring points, but smoothly connected to each other. A listing of the GEOFORM family of codes is given in the Appendix.

Figure 12: Propagation of flame front in turbulent channel flow



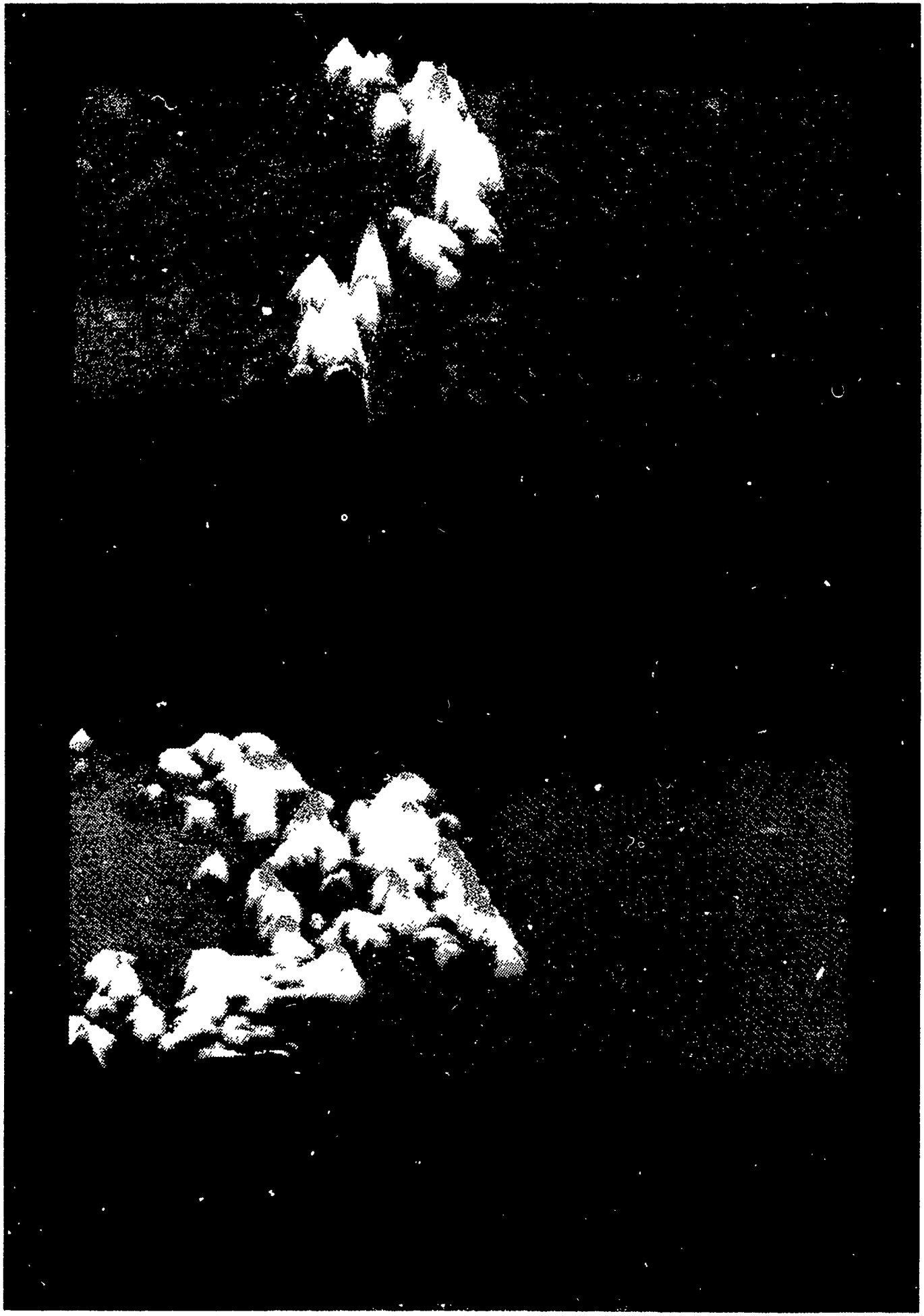


Figure 13: Propagation of flame front (late time)

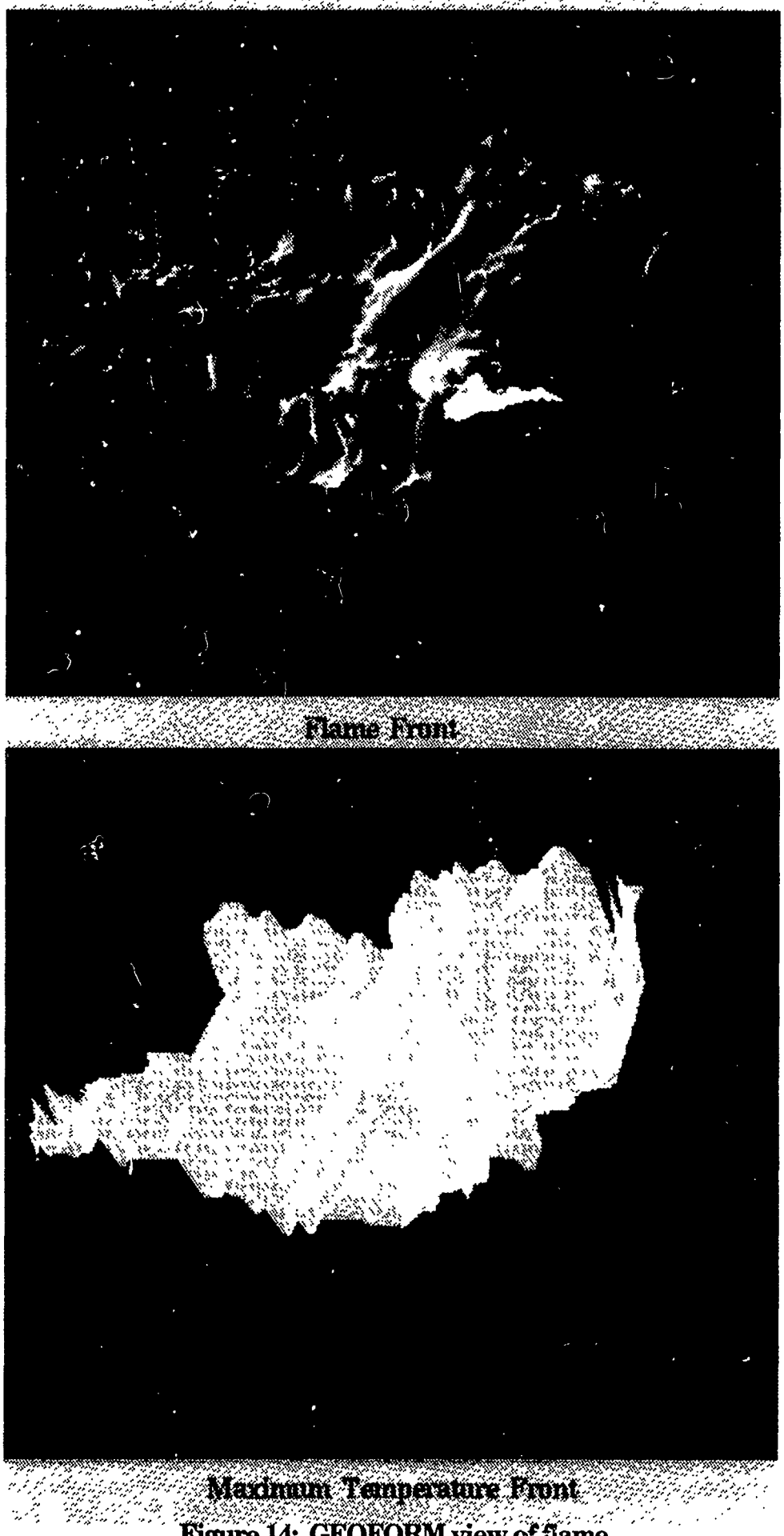


Figure 14: GEOFORM view of flame obtained using RNG-LES simulation

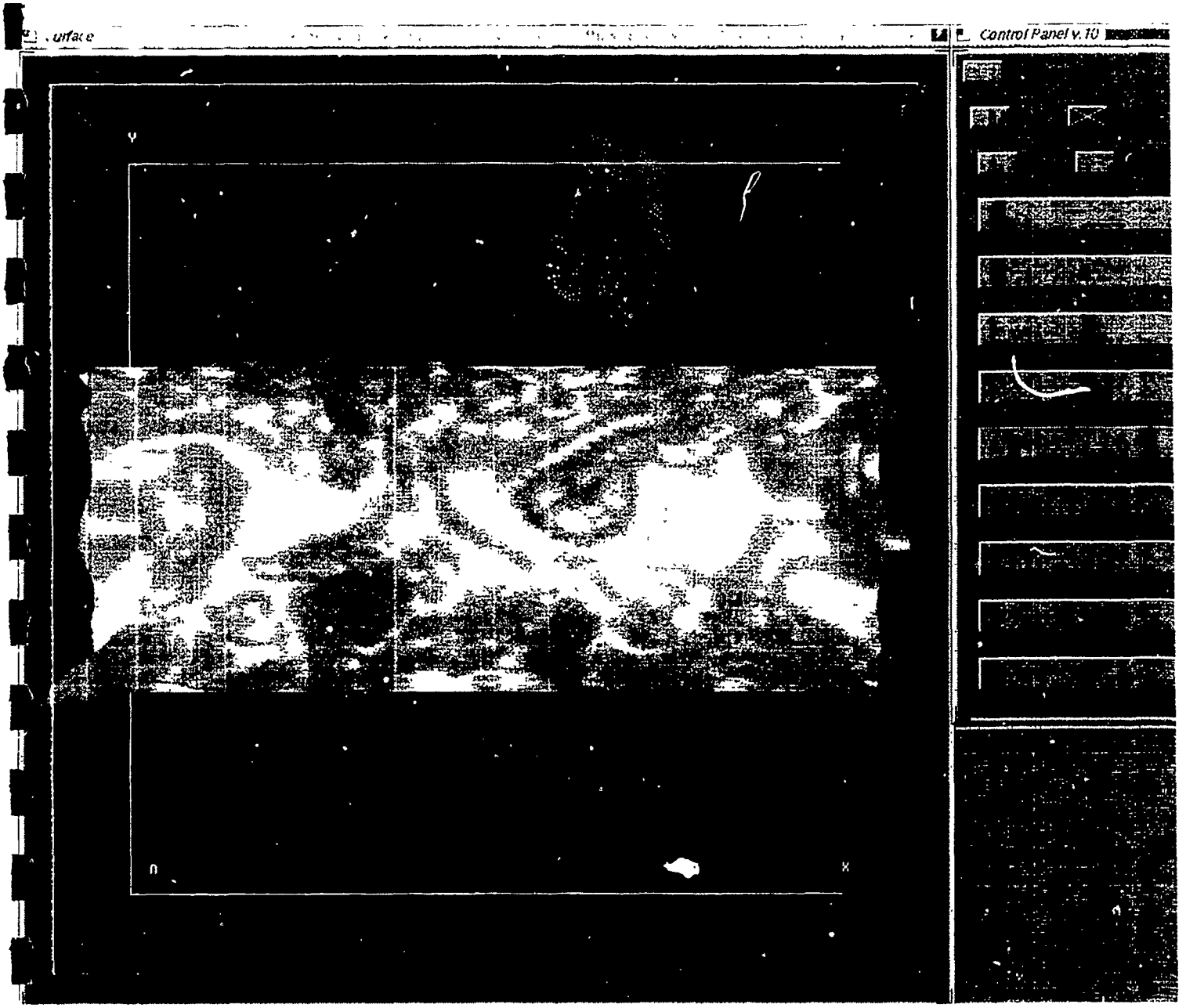


Figure 15: Head-on GEOFORM view of flame in channel

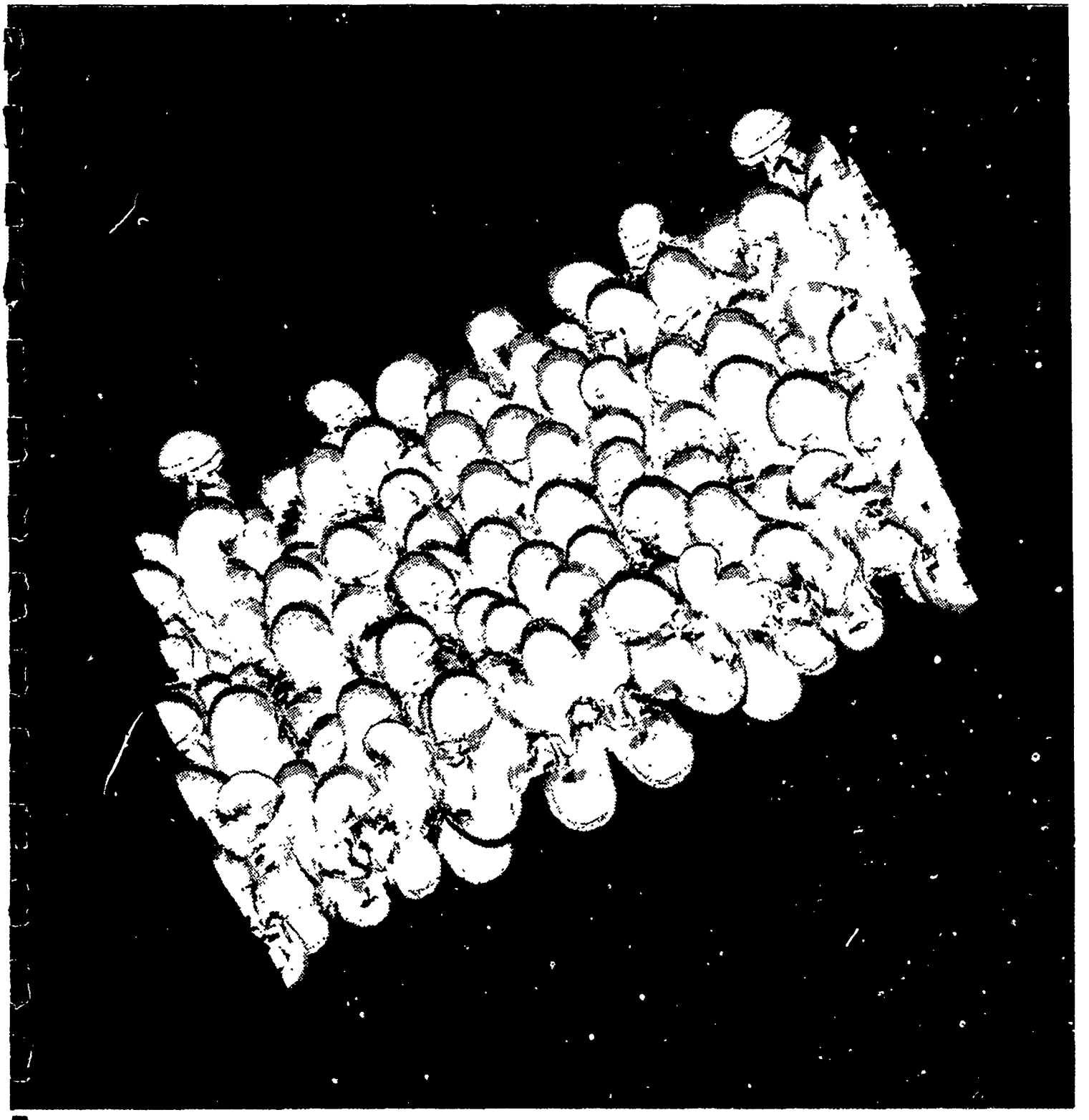


Figure 16: Three-dimensional view of random Rayleigh-Taylor instability

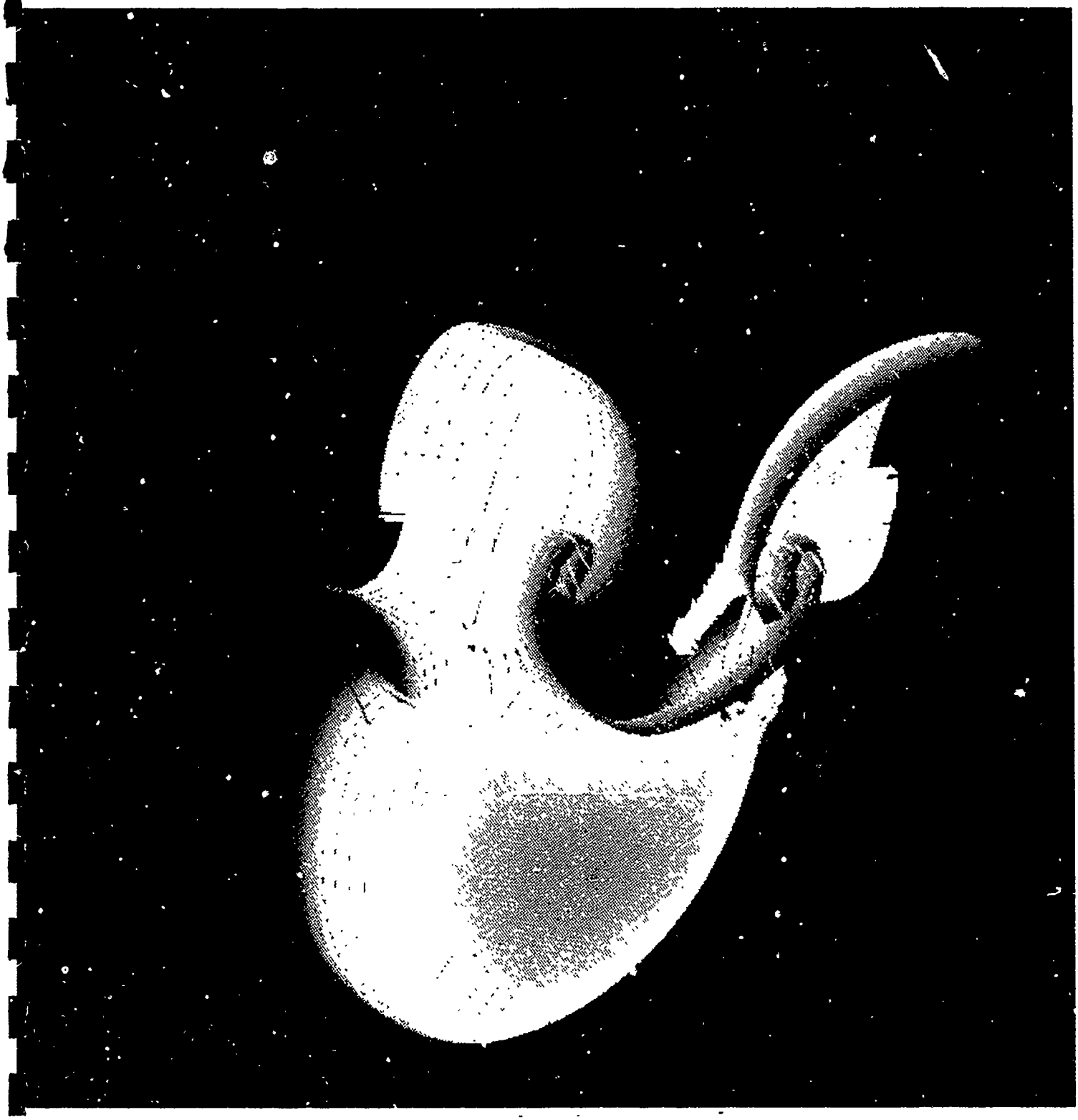


Figure 18: Three-dimensional single frequency Rayleigh-Taylor instability

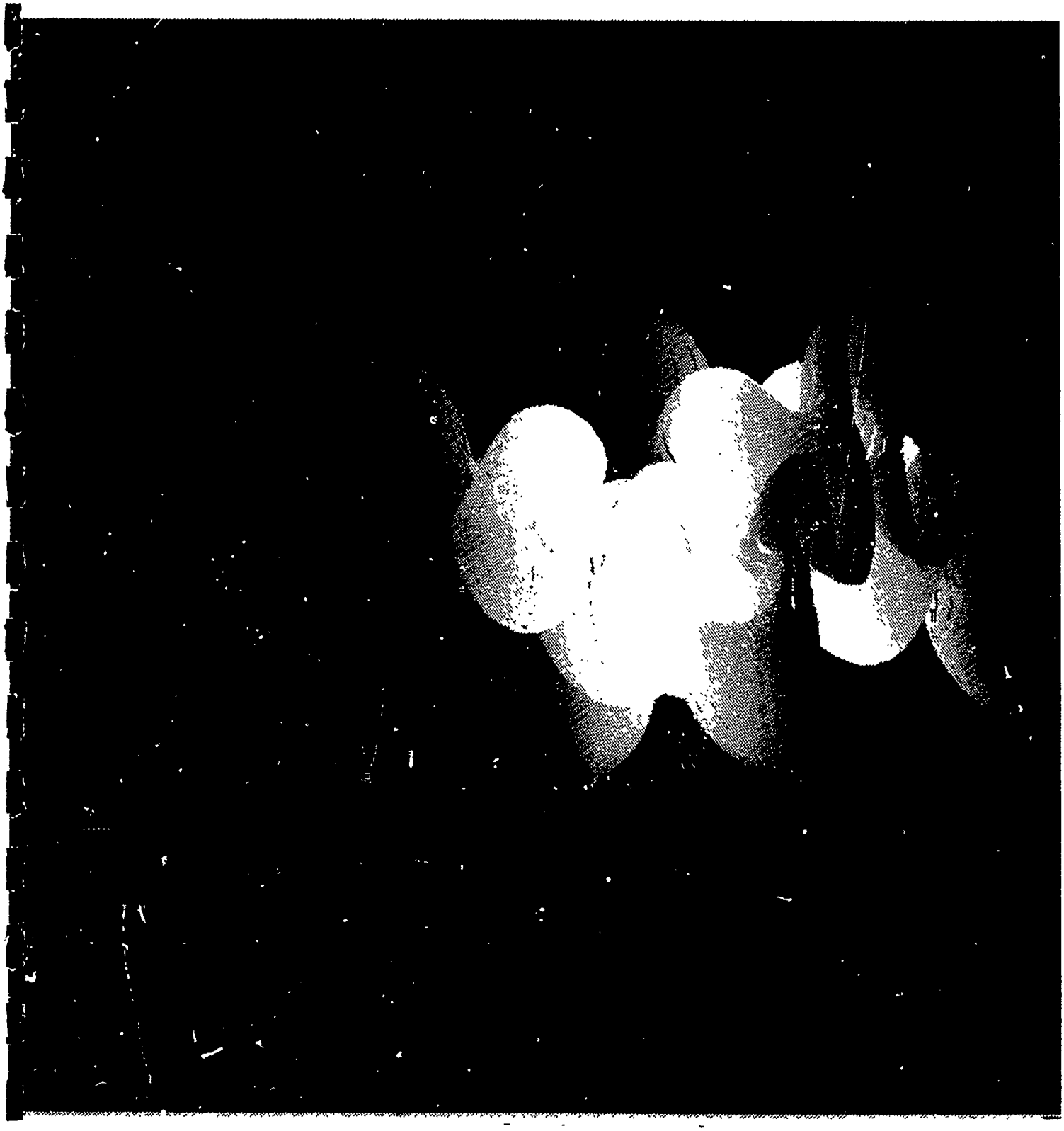


Figure 19: Early time 3D two-frequency Rayleigh-Taylor instability

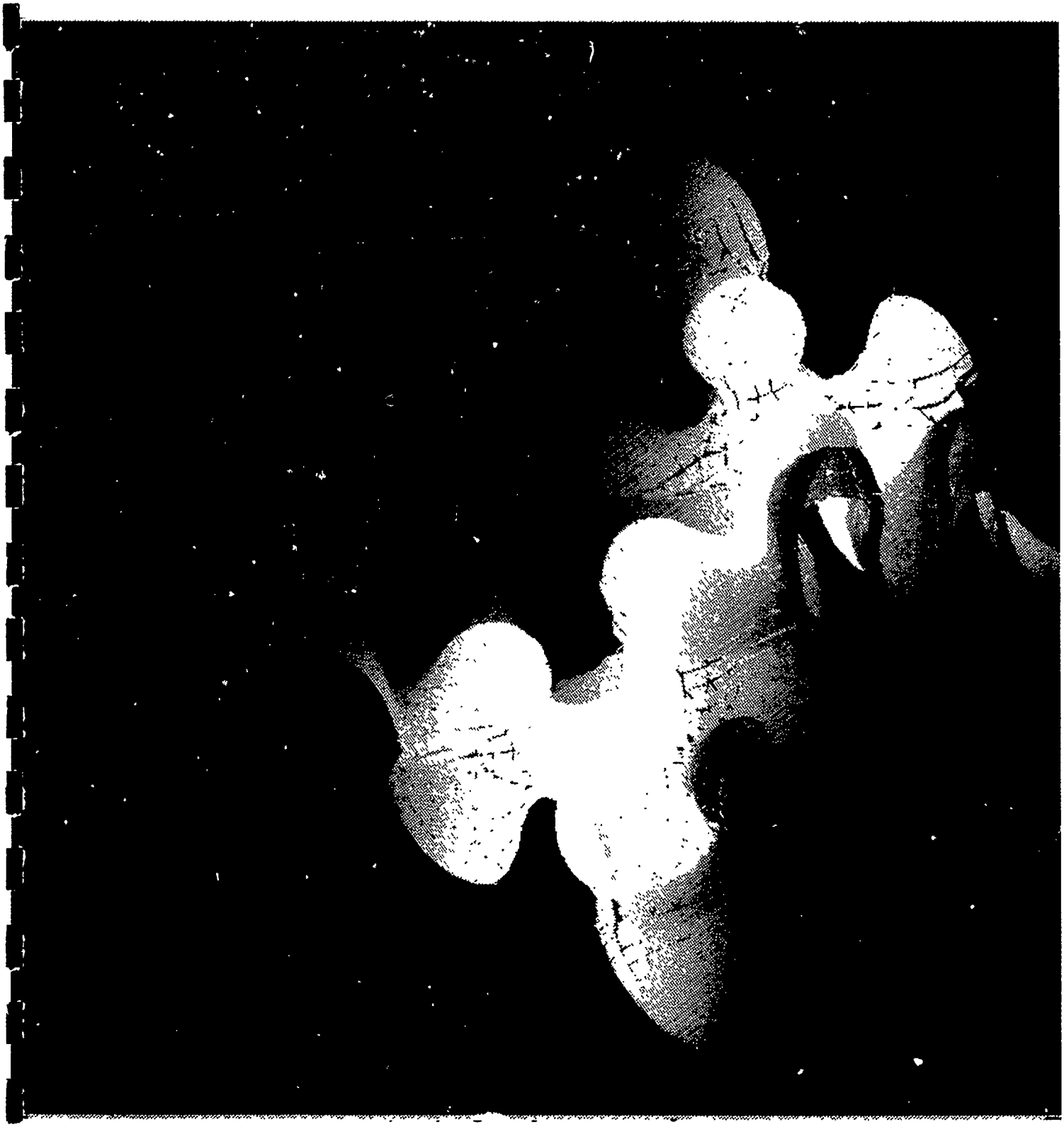


Figure 20: Late time 3D two-frequency Rayleigh-Taylor instability

ctext.c


```
#include "fmclient.h"
#include "gl.h"
#include "stdio.h"
#include <string.h>
#include<device.h>


#define JOB 1
#define EXIT 0

   float backvec2[]={0.0,0.0,0.8};
  float textvec2[]={1.0,1.0,0.0};
  int   mode ;
   long gid;

   try(text)
   char text[];

   {
   char str[100][100],str1[100];
  float size1,size2;
  FILE *fp,*fopen();
  int i,k;
  fp = fopen(text,"r");
  k = 4;
   mode = JOB;
  i=0;
  fscanf(fp,"%f %f",&size1,&size2);
  while (fgets(str1,100,fp)!= NULL)  {
   strcpy(str[i],str1);
   ++i;
   }
   foreground();
   prefposition(0,1270,0,1000);
   gid = winopen(text);
   winset(gid);
   RGBmode();
   doublebuffer();
   gconfig();
   qdevice(RIGHTMOUSE);

   while(mode) {
   while(!qtest()) {
   tryfont1(str,i,size1,size2);
   swapbuffers();
   }
   prol(str,i,size1,size2);

   }
   }


    prol(str,i,size1,size2)
    int i;
    char str[100][100];
      float size1,size2;
    {
    short val;
    int dev;
    while( qtest() ){
   switch(qread(&val) ) {
       case(RIGHTMOUSE):
       if(val ==1){
         mode = EXIT;
          winclose(gid);
          }
          break;
        case (REDRAW):
        reshapeviewport();
        while(!qtest() ) {
        tryfont1(str,i,size1,size2);
        swapbuffers();
        }
        break;
  default:
   break;
   }
   }
   }



   tryfont1(str,i,size1,size2)
    int i;
    char str[100][100];
      float size1,size2;

 {

      fmfonthandle f;
      fmfonthandle fsized;
      int x, y,k,  slen,igid,stepy;
      int beg,stop,start;
      char c;
c3f(backvec2);
      clear();
      c3f(textvec2);
     if(size1 != 0& size2 != 0) {
      beg = 1;stop = 2;start = 900;
      }
      else if(size1 !=0) {
      beg = 1;     stop = i;start =1000 -   (1000-i*100)/4;
      }

      fminit();

      f = fmfindfont("Times-Roman");

      fsized = fmscalefont(f, size1);
      fmsetfont(fsized);
      for (k = beg; k< stop; k++){
      x= 100; y=start-k*150;
      cmov2(x,y);
      fmprstr(str[k]);


      }
      stepy = (i<9) ? 100 : (start- (stop *150))/i;
    if (size2!=0) {
      if (size1 = 0)
               beg = 0;

      else
           beg = 1;
      f = fmfindfont("Times-Roman");
      fsized = fmscalefont(f, size2);
      fmsetfont(fsized);
```

```
      for (k=beg+1;k<i;k   , ,
      x= 100; y=700-(k-1)*stepy;
      cmov2(x,y);
      fmprstr(str[k]);
                  }
  }

  }
```

```
/*
 *    event.c
 *    A more rational way to handle reading the event queue
 *                Written by Wade Olsen
 */

#include <stdio.h>
#include <device.h>
#include "event.h"


typedef struct event_s
{
    int    window, device, state;
    void (*func)(void *, int);
    char *arg;
    struct event_s    *next;
} event_t, *event_p;

typedef struct update_s
{
    int *flag;
    void (*ufunc)(void *);
    char *arg;
    struct update_s *next;
} update_t, *update_p;


static event_p     event_list;
static update_p    update_list;

/*
 * This routine adds an event to be checked for to the event queue.
 * window should be the wid of the window to respond in, or ANY if
 * this event applies to all windows.  device is the device, and state
 * is the device's value (e.g.  ANY, UP, DOWN, the window id (for
 * REDRAW), etc).  Func is the function that will be called, with
 * arguments 'arg' and the device's value.
 *
 * NOTE:  the device must be queued for it to be found by the event()
 * routine-- add_event DOES NOT qdevice(device).
 */
void
add_event(window, device, state, func, arg)
int window, device, state;
void (*func)(void *, int);
char *arg;
{
    event_p new_guy;

    new_guy = (event_p)malloc(sizeof(event_t));
    new_guy->window = window;
    new_guy->device = device;
    new_guy->state  = state;
    new_guy->func   = func;
    new_guy->arg    = arg;
    new_guy->next   = event_list;
    event_list = new_guy;
}


/*
 *       Specify a function to be called if there is nothing in the queue
 * and (*flag) is non-zero.  If no update function is active, or
 * (*flag) is 0, then event() will block on a qread.  If there is an
 * active update function, event() will continuously call the update
 * function, hogging the cpu.
 */
void
add_update(flag, ufunc, arg)
int *flag;
void (*ufunc)(void *);
char *arg;
{
    update_p    new_guy;

    new_guy = (update_p)malloc(sizeof(update_t));
    new_guy->flag = flag;
    new_guy->ufunc = ufunc;
    new_guy->arg  = arg;
    new_guy->next = update_list;
    update_list = new_guy;
}

/*
 * The main Event.  Call this repeatedly to automagically handle
 * reading the queue, and calling your functions to handle what
 * appears there.
 */
void
event()
{
    void find_event(void), event_inputchange(void);
    int find_update(void);
    static int initialized = 0;

    if (initialized == 0)
    {
        add_event(ANY, INPUTCHANGE, ANY, event_inputchange, NULL);
        qdevice(INPUTCHANGE);
        initialized = 1;
    }

    /* Something in the queue?  Handle it */
    if (qtest())
        find_event();
    /*
     * Or, if there's no update function, wait for something to appear
     */
    else if (find_update() == 0)
        find_event();
}

int
find_update()
{
    update_p    scan;
    int         updated = 0;

    for (scan = update_list; scan && updated == 0; scan = scan->next)
    {
        if (*scan->flag)
        {
            (*scan->ufunc)(scan->arg);
            updated = 1;
        }
    }

    return(updated);
```

```
}

int context, state, device;

void
event_inputchange()
{
    context = winget();
}

void
find_event()
{
    event_p scan;
    short   s;

    device = qread(&s);
    state = s;
    for (scan = event_list; scan; scan = scan->next)
    {
        if ( ((scan->window == ANY) || (context == scan->window))
            && ((scan->device == ANY) ||(device == scan->device))
            && ((scan->state == ANY) || (state == scan->state)))
        {
            (*scan->func)(scan->arg, state);
        }
    }
}
```

```
#include <stdio.h>
#include <gl/gl.h>
#include <string.h>
#include <fmclient.h>
#include <math.h>
#include <forms.h>
#include "display.h"
#include "light.h"
#include "cpath.h"


#define maxcol 256
#define MATRIXSIZE 16                /*size of matrix in the stack*/
#define LIGHTINDEX 10               /*size of sun file*/
#define   k      3.5
#define   ANY    -1
#define   SIZE   50000


float   backvec[] = {0.0, 0.0, 0.0} ;
float   blackvec[] = {0.0, 0.0, 0.0} ;
float   redvec[] = {1.0, 0.0, 0.0} ;
float   whitevec[] = {1.0, 1.0, 1.0} ;
float   yelvec[] = {1.0,1.0,0.0};

float  color_palette[maxcol][3];
int ncol;
Object Vobj,sphere,Vobj1;
float wx2,wy2,wz2,scal_par;
typedef float   Vec[3] ;

typedef struct {
        int     p_nsides ;
        Vec     ** p ;
        int      *s ;
} Polygon ;

 int    negflag ;


float idmat[] = {1.0, 0.0, 0.0, 0.0,
                 0.0, 1.0, 0.0, 0.0,
                 0.0, 0.0, 1.0, 0.0,
                 0.0, 0.0, 0.0, 1.0} ;
float OldMatrix[16],NewMatrix[16];

Polygon *
MakePolygon(nsides)
 int nsides ;
{
        Polygon * p ;

        p = (Polygon *) malloc (sizeof(Polygon)) ;

        p -> p_nsides = nsides ;

        p -> p = (Vec **) calloc(nsides, sizeof(Vec *)) ;
        p -> s = (int * ) malloc(3* sizeof(int )) ;

        return p ;
}


Vec     V[SIZE] ;
Polygon * P[SIZE] ;

Vec     N[SIZE],N1[SIZE] ;
int     npoints, npolys ;
Vec     mins = {1000000, 1000000, 1000000} ;
Vec     maxs = {-1000000, -1000000, -1000000} ;
float   V1[SIZE] ,V2[SIZE][3];

#define min(a, b) ((a) < (b) ? (a) : (b))
#define max(a, b) ((a) > (b) ? (a) : (b))

float Max ,scal;

        float shiftx ,shifty,shiftz ;
ReadObject()
{

        int i, j, ku,nverts, index[3], nsides ;
        Vec     tmp1, tmp2 ;

        scanf("%d %d\n", & npolys, & npoints) ;

/*
        fprintf(stderr, "%d points, %d polys\n", npoints, npolys) ;
        */
                for (j = 0 ; j < 3 ; j++) {
                        mins[j] =  1e10;
                        maxs[j] =   -1e10;
                }

        for (i = 0 ; i < npoints ; i++) {
                scanf("%f %f %f\n", &(V[i][0]), &(V[i][1]), &(V[i][2])) ;
        }

/*Lala: I changed here */
        for(i = 0;i<npoints;i++)
                for (ku = 0 ; ku < 3 ; ku++) {
                        mins[ku] = min(mins[ku], V[i][ku]) ;
                        maxs[ku] = max(maxs[ku], V[i][ku]) ;
                    }
        Max = max(maxs[0]-mins[0], maxs[1]-mins[1]) ;
        Max = max(Max, maxs[2]-mins[2]) ;
        scal=ncol/(maxs[2]-mins[2]);
        shiftx = 0.5*(maxs[0]+mins[0]);
        shifty = 0.5*(maxs[1]+mins[1]);
        shiftz = 0.5*(maxs[2]+mins[2]);


        for (i = 0; i < npoints ; i++) {
        V[i][0] = V[i][0]- shiftx;
        V[i][1] = V[i][1] - shifty;
        V[i][2] = V[i][2] -  shiftz;

        }
        maxs[2] = maxs[2]-shiftz;
        mins[2] = mins[2]-shiftz;



        for (i = 0 ; i < npolys ; i++) {

                P[i] = MakePolygon(3) ;
                for (j = 0 ; j < 3 ; j++) {
                        scanf("%d", &index[j]) ;
                        P[i]->s[j] = index[j] -1 ;
                        P[i]->p[j] = V + (index[j] - 1) ;
```

```
                        V2[i][j] = V[index[j] -1][2];
                }
           V1[i] = V2[i][0] + V2[i][1] +V2[i][2];
           V1[i] = V1[i]/3.;
        }
/*

        fprintf(stderr, "bbx: %f - %f\n", mins[0], maxs[0]) ;
        fprintf(stderr, "bby: %f - %f\n", mins[1], maxs[1]) ;
        fprintf(stderr, "bbz: %f - %f\n", mins[2], maxs[2]) ;
        */
                /*mins[2]=0;*/

        scal_par = Max/2.0;


        for (i = 0 ; i < npolys ; i++) {
                VecSub(P[i]->p[0], P[i]->p[1], tmp1) ;
                VecSub(P[i]->p[2], P[i]->p[1], tmp2) ;
                VecCross(tmp1, tmp2, N[i]) ;
                VecNormalize(N[i]) ;
                if (negflag) {
                  if(N[i][0] < 0.)N[i][0] = - N[i][0];
                  if(N[i][1] < 0.)N[i][1] = - N[i][1];
                  if(N[i][2] < 0.)N[i][2] = - N[i][2];

                        }
             else{
                N[i][0] = - N[i][0] ;
                        N[i][1] = - N[i][1] ;
                        N[i][2] = - N[i][2] ;
                }
        }
        for (i = 0; i < npolys ; i++) {
        N1[i][0] = 0.0;
        N1[i][1] = 0.0;
        N1[i][2] = 0.0;
                    }

        for (i = 0; i < npolys ; i++) {
        for (j = 0; j< 3;j++) {
        N1[ P[i]->s[j] ][0]+= N[i][0];
        N1[ P[i]->s[j] ][1]+= N[i][1];
        N1[ P[i]->s[j] ][2]+= N[i][2];
        }
          }
        for (i = 0; i < npolys ; i++)
         VecNormalize(N1[i]);
}

VecNegate(a)
 Vec a ;
{
        a[0] = 0 - a[0] ;
        a[1] = 0 - a[1] ;
        a[2] = 0 - a[2] ;
}

VecSub(a, b, c)
 Vec a, b, c ;
{
        c[0] = a[0] - b[0] ;
        c[1] = a[1] - b[1] ;
        c[2] = a[2] - b[2] ;
}

VecCross(a, b, c)
 Vec a, b, c ;
{
        c[0] = a[1] * b[2] - b[1] * a[2] ;
        c[1] = - (a[0] * b[2] - b[0] * a[2]) ;
        c[2] = a[0] * b[1] - b[0] * a[1] ;
}

VecNormalize(a)
 Vec a ;
{
        float l ;
        l = (float) sqrt(a[0] * a[0] + a[1] * a[1] + a[2] * a[2]) ;

        a[0] /= l ;
        a[1] /= l ;
        a[2] /= l ;
}

float shinymaterial[] = { SPECULAR, 1.0, 0.2, 0.8,
                          EMISSION , 0.8,0.8,0.0,
                            DIFFUSE, 0.8, 0.0, 0.0,
                            AMBIENT, 0.2, 0.0, 0.0,
                            SHININESS, 3.0,
                            LMNULL } ;
float sun1[] = { LCOLOR,1.0, 1.0,1.0,
                 POSITION, 0.0, 0.0, 1.0,0.0,
                 LMNULL } ;

float sun2[] = { LCOLOR,1.0, 0.0,0.0,
                 POSITION, 0.0, 0.0, 1.0,0.0,
                 LMNULL } ;

float sun3[] = { LCOLOR,0.0, 0.0,1.0,
                 POSITION, 0.0, 0.0, 1.0,0.0,
                 LMNULL } ;


  float kvadrat1[] = {-0.5,-0.5,-0.5};
  float kvadrat2[] = {-0.5,0.5,-0.5};
  float kvadrat3[] = {0.5,0.5,-0.5};
  float kvadrat4[] = {0.5,-0.5,-0.5};
  Drawbox()
  {
  char string1[5],string2[5],string3[5],string4[5];

  sprintf(string1,"%s","X");
  sprintf(string2,"%s","Y");
  sprintf(string3,"%s","Z");
  sprintf(string4,"%s","O");
   c3f(whitevec);


  cmov(0.50,-0.47,-0.50);
  charstr(string1);
  cmov(-0.50,0.53,-0.50);
  charstr(string2);
  cmov(-0.53,-0.53,0.53);
  charstr(string3);
  cmov(-0.47,-0.47,-0.47);
  charstr(string4);
```

```
   pushmatrix();
   bgnclosedline();
   v3f(kvadrat1);
   v3f(kvadrat2);
   v3f(kvadrat3);
   v3f(kvadrat4);
   endclosedline();
   popmatrix();
   pushmatrix();
   translate(0.0,0.0,1.0);
   bgnclosedline();
   v3f(kvadrat1);
   v3f(kvadrat2);
   v3f(kvadrat3);
   v3f(kvadrat4);
   endclosedline();
   popmatrix();

   pushmatrix();
   rotate(-900,'y');
   bgnclosedline();
   v3f(kvadrat1);
   v3f(kvadrat2);
   v3f(kvadrat3);
   v3f(kvadrat4);
   endclosedline();
   popmatrix();
}

init_palette()
{
int l,j;
float dxl;
for(l = 0;l<3;l++)
for(j = 0;j<ncol;j++)
color_palette[j][l] =0.0;

 for(l = 0;l<3;l++){
dxl= 4./ncol;
   if(l == 0) {
     for(j = ncol;j>= (int)(3*ncol/4)+1;j--){
      color_palette[j][l] = 1.0;
     }
     for(j = (int)(3*ncol/4);j>=ncol/2;j--){
      color_palette[j][l] = dxl*(j -(int)(ncol/2));
     }
   }
   if(l == 1) {
         for(j =(int) (ncol); j>= (int)(3*ncol/4);j--){
       color_palette[j][l] = 1 - dxl*(j -(int)( 3*ncol/4));
      }
       for(j =(int) (3*ncol/4); j>= (int)(ncol/4);j--){
        color_palette[j][l] = 1.0;
       }
       for(j =(int) (ncol/4); j>= 0;j--){
         color_palette[j][l] =  dxl*j;
    }

    }
   if(l == 2){
     for(j = ncol/2;j>= (int)(ncol/4);j--) {
       color_palette[j][l] =1.0 - dxl*(j-ncol/4);
      /* printf("%f %d %d \n", color_palette[j][l],l,j);*/
     }
     for(j = (int)(ncol/4);j>=0 ;j--) {
       color_palette[j][l] = 1.0;
     }
   }
 }
}


 get_palette()
{
int j;
FILE *fp, *fopen();
fp = fopen("color_palette","r");

  if(fp != NULL){
    fscanf(fp,"%d",&ncol);
      for(j = 0;j<ncol;j++)
       fscanf(fp,"%f %f %f",&color_palette[j][0],&color_palette[j][1],
              &color_palette[j][2]);


 fclose(fp);
scal=ncol/(maxs[2]-mins[2]);
    }
   else {ncol = maxcol; init_palette();}

}

get_surface()
{

FILE *fp, *fopen();


fp = fopen("lighting","r");
if(fp !=NULL) {
fscanf(fp,"%f %f %f",&shinymaterial[1], &shinymaterial[2], &shinymaterial[3]);
fscanf(fp,"%f %f %f",&shinymaterial[5],&shinymaterial[6],&shinymaterial[7]);
fscanf(fp,"%f %f %f",&shinymaterial[9],&shinymaterial[10],&shinymaterial[11]);
fscanf(fp,"%f %f %f",&shinymaterial[13],&shinymaterial[14],&shinymaterial[15])
;
fscanf(fp,"%f %f %f",&sun1[1],&sun1[2],&sun1[3]);
lmdef(DEFMATERIAL, 1, 19, shinymaterial) ;
lmbind(MATERIAL, 1) ;
lmdef(DEFLIGHT,1, 10,sun1);
lmbind(LIGHT0, 1);
}
fclose(fp);
}

save_position()
{
FILE *fp,*fopen();
int i;
fp = fopen("pnl_position","w");
for(i = 0;i< MATRIXSIZE;i++)
fprintf(fp,"%f ",OldMatrix[i]);
for(i = 0;i< LIGHTINDEX;i++)
fprintf(fp,"%f ",sun1[i]);
for(i = 0;i< LIGHTINDEX;i++)
fprintf(fp,"%f ",sun2[i]);
for(i = 0;i< LIGHTINDEX;i++)
fprintf(fp,"%f ",sun3[i]);
fclose(fp);
```

```
}


/*get_position()
{
int i;
FILE *fp,*fopen();
fp = fopen("pnl_position","r");
if(fp != NULL) {
   for(i = 0; i< MATRIXSIZE; i++)
     fscanf(fp,"%f ",&(OldMatrix[i]));
 for(i = 0;i< LIGHTINDEX;i++)
 fscanf(fp,"%f ",&(sun1[i]));
 for(i = 0;i< LIGHTINDEX;i++)
 fscanf(fp,"%f ",&(sun2[i]));
 for(i = 0;i< LIGHTINDEX;i++)
 fscanf(fp,"%f ",&(sun3[i]));
 fscanf(fp,"%f %f %f ",&(light1->val),&(light2->val),&(light3->val));
fclose(fp);
lmdef(DEFLIGHT,1, 10,sun1);
lmdef(DEFLIGHT,2, 10,sun2);
lmdef(DEFLIGHT,3, 10,sun3);
if(light1->val)lmbind(LIGHT0, 1) ;else lmbind(LIGHT0,0);
if(light2->val)lmbind(LIGHT1, 2) ;else lmbind(LIGHT1,0);
if(light3->val)lmbind(LIGHT2, 3) ;else lmbind(LIGHT2,0);
loadmatrix(OldMatrix);

}

}
*/


long igid;


init_mywin()
{

  keepaspect(1, 1) ;
        foreground() ;
        igid = winopen("surface") ;
        RGBmode() ;
        doublebuffer() ;
        gconfig() ;

        lsetdepth(0, 0x7FFFFF) ;
        zbuffer(TRUE) ;

        mmode(MVIEWING) ;




    Set_Slider_Bounds()


 fl_set_slider_bounds(xrot,-1800,1800);
 fl_set_slider_bounds(yrot,-1800,1800);
 fl_set_slider_bounds(zrot,-1800,1800);
 fl_set_slider_bounds(shix,-1.0 *Max,Max);
 fl_set_slider_bounds(shiy,-1.0 *Max,Max);
 fl_set_slider_bounds(shiz,-1.0 *Max,Max);
 fl_set_slider_bounds(xsc,0.1, 10);
 fl_set_slider_bounds(ysc,0.1, 10);
 fl_set_slider_bounds(zsc,0.1, 10);
 fl_set_slider_bounds(size,0.5, 5.0);
}

Set_Slider_Values()
{
fl_set_slider_value(xrot,0.0);
fl_set_slider_value(yrot,0.0);
fl_set_slider_value(zrot,0.0);
fl_set_slider_value(shix,0.0);
fl_set_slider_value(shiy,0.0);
fl_set_slider_value(shiz,0.0);
fl_set_slider_value(xsc,1.0);
fl_set_slider_value(ysc,1.0);
fl_set_slider_value(zsc,1.0);
fl_set_slider_value(size,2.0/Max);
}

#define GETVAL(x) fl_get_slider_value(x)


 void Change_Stack(FL_OBJECT *obj,long arg)
{
printf("I got in Change_Stack\n");
callobj(Vobj1);
if(obj == xrot )rotate((short)GETVAL(xrot) ,'x');
if(obj == yrot )rotate((short)GETVAL(yrot) ,'y');
if(obj == zrot )rotate((short)GETVAL(zrot) ,'z');
if(obj == xsc || obj == ysc || obj == zsc) scale(GETVAL(xsc),GETVAL(ysc),GETVAL(zsc));
if(obj == size) scale(GETVAL(size),GETVAL(size),GETVAL(size));
if(obj == shix || obj == shiy || obj == shiz) translate( GETVAL(shix),
GETVAL(shiy), GETVAL(shiz));
multmatrix(OldMatrix);
getmatrix(OldMatrix);
}

MakeVobj()
{
makeobj(Vobj1 = genobj() );
       loadmatrix(idmat) ;
        ortho(-2.0,2.0,-2.0,2.0,-2.0,2.0);
        closeobj();
}


MakeLight1()
{
sun1[5] = GETVAL(xlight1);
sun1[6] = GETVAL(ylight1);
sun1[7] = GETVAL(zlight1);
lmdef(DEFLIGHT, 1, 10, sun1) ;
lmbind(LIGHT0, 1) ;
sun2[5] = GETVAL(xlight2);
sun2[6] = GETVAL(ylight2);
sun2[7] = GETVAL(zlight2);
lmdef(DEFLIGHT, 2, 10, sun2) ;
lmbind(LIGHT1, 2) ;
sun3[5] = GETVAL(xlight3);
sun3[6] = GETVAL(ylight3);
```

```
sun3[7] = GETVAL(zlight3);
lmdef(DEFLIGHT, 3, 10, sun3) ;
lmbind(LIGHT2, 3) ;

}

void save_picture(FL_OBJECT *obj,long arg)
{
char savePict[50],*out_rgb;
long xor,yor,xsize,ysize;

winset(igid);
getorigin(&xor,&yor);
getsize(&xsize,&ysize);

out_rgb = fl_get_input(obj);
sprintf(savePict,"%s%s %d %d %d %d ","scrsave ",out_rgb,xor,
xor+xsize,yor,yor+ysize);

system(savePict);
}


void make_reset(FL_OBJECT *obj , long arg)
{
callobj(Vobj1);
scale(2.0/Max,2.0/Max,2.0/Max);
getmatrix(OldMatrix);
}



 void DrawObj(FL_OBJECT *obj, long arg)
{
   static     FL_OBJECT *CurObj;

        int i, j,col_ind ;
        if(obj ==  map || obj == model || obj == surface || obj ==
smooth ) CurObj = obj;
        get_surface();
        winset(igid);
        reshapeviewport();
        c3f(backvec);
        clear();
        zclear();
        MakeVobj();
        callobj(Vobj1);
         MakeLight1;
         if(obj == box){
         pushmatrix();
         scale(2.0,2.0,2.0);
         Drawbox();
         popmatrix();
       }
                if (CurObj == model)  {
mbind(MATERIAL, 0) ;
mbind(LIGHT0, 0) ;
mbind(LIGHT1, 0) ;
mbind(LIGHT2, 0) ;
mbind(LMODEL, 0) ;
        for (i = 0 ; i < npolys ; i++) {
                c3f(redvec) ;
                bgnclosedline() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                        v3f(P[i]->p[j]) ;
                }
                endclosedline() ;

}
}
  if (CurObj == map)  {
lmbind(MATERIAL, 0) ;
lmbind(LIGHT0, 0) ;
lmbind(LIGHT1, 0) ;
lmbind(LIGHT2, 0) ;
lmbind(LMODEL, 0) ;

        for (i = 0 ; i < npolys ; i++) {
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                col_ind=(int)((V2[i][j]-mins[2])*scal);
                col_ind=min(ncol-1,max(0,col_ind));
                c3f(&(color_palette[ col_ind][0])) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
        }
if(CurObj == surface|| CurObj == smooth)   {
lmbind(MATERIAL, 1) ;
lmbind(LIGHT0, 1) ;
lmbind(LIGHT1, 2) ;
lmbind(LIGHT2, 3) ;
lmbind(LMODEL, 1) ;
        for (i = 0 ; i < npolys ; i++) {
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                c3f(yelvec ) ;
                        if(CurObj == surface) n3f(N[i]) ;
                        if(CurObj == smooth) n3f(N1[P[i]->s[j]]) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
        }


        swapbuffers() ;
      }



void show_cedit (FL_OBJECT *obj, long arg)
{
   char * str[100];
   sprintf(str,"%s%s %d %s",PATH,"mycedit",ncol,"&");
   system(str);
}

main(argc,argv)
int argc;
char *argv[];
{

       FILE *fp, *fopen();
       FL_OBJECT *obj;
        int i,j;
        short  val ;
```

```
        int dev;
        ncol = maxcol;

        negflag = 0;
        if (argc == 2)
                negflag = 1 ;
        init_palette();
        ReadObject() ;

        init_mywin();
        create_the_display();
        create_the_light();
        Set_Slider_Bounds();
        Set_Slider_Values();
        fl_show_form(DISPLAY,FL_PLACE_SIZE,TRUE,"display");
         DrawObj();
        while (obj != quit_disp) {
          obj = fl_do_forms();
          DrawObj(obj,0);
         }
}
```

```
#include<stdio.h>
#include <forms.h>
#include "geoplane.h"
#include "option_form.h"
#include <string.h>
#include "cpath.h"
#include "sandwatch"


#define RUN 1
#define TRIANGULATION 2
#define INTERPOLATION 3
#define DATA_EDITOR  4
#define DISPLAY 5




int norm, peak,hardcopy;
float wlog;
char hardstr[100];
char  filename[50], runinfo[50],trianginfo[50],showinfo[50];
char leadersur[50],  surfaceinfo[50],intinfo[50],geomview[50];
char text[50] ,cleanstr[50];
FILE *fp1,*fp2, *fp, *fopen();
int max_size;
int menu2;
int get_max_size();
char stroka[100];

 void show_info(FL_OBJECT *obj, long arg){
fl_show_message("GEOFORM Version 2.0 ", "Cambridge Hydrodynamics Inc.","");
}


initialize(){

    sprintf(runinfo,"%s%s",PATH,"run.info");
    sprintf(trianginfo,"%s%s",PATH,"triang.info");
    sprintf(showinfo,"%s%s",PATH,"show.info");
    sprintf(intinfo,"%s%s",PATH,"int.info");
    sprintf(surfaceinfo,"%s%s",PATH,"surface.info");
    sprintf(geomview,"%s%s%s",PATH,"geomview",PLACE);
    sprintf(text,"%s%s",PATH,"text1");
    sprintf(leadersur,"%s%s",PATH,"leader.sur");
    sprintf(cleanstr,"%s%s",PATH,"clean");
    system(cleanstr);
  fp1=fopen(leadersur,"r");
   fp2 = fopen("leader","w");
   while(fgets(stroka,100,fp1)!=NULL)
    fprintf(fp2,"%s",stroka);


    fclose(fp1);
    fclose(fp2);

   wlog = -1;
   peak = -1;
   norm = 0;
   hardcopy = 0;
   }
```

```
myMenus()
{
menu2 = defpup("ADD POINTS%t %F|2*size of data %x2| 4*size of data %x4| 8 *size of data %
x8",get_max_size);
}

void choice(FL_OBJECT *obj, long arg)
{
if(arg == RUN)
    {
      max_size = dopup(menu2);
      change_cursor();
      start_(filename,&norm,&wlog,&peak);
      trian_(&hardcopy);
      if(max_size> 1){
       add_(&max_size);
       int_(&max_size);
      if(hardcopy)system(hardstr);
      }
      restore_cursor();
      fl_show_message("DATA IS PROCESSED","PRESS SAVE BUTTON  ","TO SAVE OUTPUT IN THE FI
LE");
     }

if(arg == TRIANGULATION)
    {
      change_cursor();
      start_(filename,&norm,&wlog,&peak);
      trian_(&hardcopy);
restore_cursor();
      fl_show_message("TRIANGULATION IS DONE","","");
    }
if(arg == INTERPOLATION)
    {
      max_size = dopup(menu2);

      if((fp1 = fopen("triang","r")) == NULL) {
       fl_show_message("YOU HAVE TO RUN","TRIANGULATION" ,"FIRST");
       fclose(fp1);
      }
      else{
      change_cursor();
       add_(&max_size);
       int_(&max_size);
       if(hardcopy)system(hardstr);
restore_cursor();
       fl_show_message("INTERPOLATION","IS ","DONE");
      }
    }
 if(arg == DATA_EDITOR){
      start_(filename,&norm,&wlog,&peak);
      show_(text,&hardcopy);
      if(hardcopy)system(hardstr);
    }
 if(arg == DISPLAY){
      change_cursor();
     system(geomview);
      printf("%s\n", geomview);
restore_cursor();
}
}
```

```
void save_triang(FL_OBJECT *obj, long arg)
{
char locstr[100];
sprintf(locstr,"%s%s%s","launch -h \"cp triang\" -m \"Save file as\" -c \"",
 filename,".tri\"");
system (locstr);

}


void get_menu(FL_OBJECT *obj, long arg)
{
fl_show_form(option_form,FL_PLACE_SIZE, FALSE, NULL);
}

int quit = 0;
void make_quit(FL_OBJECT *obj, long arg)
{
quit = 1;
}



void normalize(FL_OBJECT *obj, long arg)
{
norm = 1;
}

void logar(FL_OBJECT *obj, long arg)
{
wlog = 8;
}

void kill_peak(FL_OBJECT *obj, long arg)
{
peak =  fl_show_question("ERRORS IN THE FILE?","","");
}

void reset(FL_OBJECT *obj, long arg)
{
wlog = -1;
peak = -1;
norm = 0;
hardcopy = 0;

}


get_max_size(n)
int n;
{
max_size = n;
return n;



void show_files(FL_OBJECT *obj, long arg)

char stroka[100],*fname;
system(cleanstr);
name = fl_show_file_selector("INPUT FILE NAME","","","");
   if(fname == NULL || fname[0] == '\0')return;
else
   strcpy(filename,fname);

  fp1=fopen(leadersur,"r");
   fp2 = fopen("leader","w");

   while(fgets(stroka,100,fp1)!=NULL) {
    fprintf(fp2,"%s",stroka);
    }
    fclose(fp1);
    fclose(fp2);
    strcpy(fnameobj->label, filename);
  fl_redraw_object(fnameobj);
}

make_hardcopy()
{

hardcopy = 1;
sprintf(hardstr,"%s%s%s",
"launch -h \"mv hardtmp\" -m \"Save hardcopy as\"  -c \"",
filename,".mongo\"");

}

void get_out( FL_OBJECT *obj, long arg)
{
fl_hide_form(option_form);
}



change_cursor()
{
curstype(C16X1);
defcursor(1,sandwatch_bits);
curorigin(1,8,8);
setcursor(1,0,0);
}

restore_cursor()
{
setcursor(0,0,0);
}


main(argc,argv)
int argc;
char *argv[];
{
FL_OBJECT *obj;
char *fname;
if(argc < 2){
 fname = fl_show_file_selector("INPUT FILE NAME","","","");
 if(fname != NULL && fname[0] != '\0')
sprintf(filename,"%s",fname);
}
else
  strcpy(filename,argv[1]);
```

```
initialize();
create_the_forms();
create_the_options();
strcpy(fnameobj->label, filename);
fl_redraw_object(fnameobj);
fl_show_form(GEOPLANE,FL_PLACE_SIZE,TRUE, NULL);
   myMenus();
while (obj != exit_but){
obj = fl_do_forms() ;
}
fl_hide_form(GEOPLANE);
system(cleanstr);
}
```

ftext.c

```
#include "fmclient.h"
#include "gl.h"
#include "stdio.h"
#include <string.h>
#include<device.h>


#define JOB 1
#define EXIT 0

   float backvec3[]={0.0,0.0,0.8};
  float textvec3[]={1.0,1.0,0.0};
  int   mode ;
   long gid;

   tryl_(text)
   char text[];

   {
  char str[100][100],str1[100];
  float size1,size2;
  FILE *fp,*fopen();
  int i,k;
  fp = fopen(text,"r");
  k = 4;
   mode = JOB;
  i=0;
  fscanf(fp,"%f %f",&size1,&size2);
  while (fgets(str1,100,fp)!= NULL)  {
   strcpy(str[i],str1);
   ++i;
   }
   foreground();
   prefposition(0,1270,0,1000);
   gid = winopen(text);
   winset(gid);
   RGBmode();
   doublebuffer();
   gconfig();
   c3f(backvec3);
   clear();
   qdevice(RIGHTMOUSE);

   while(mode) {
   while(!qtest()) {
   tryfont1(str,i,size1,size2);
   swapbuffers();
   }
   prol(str,i,size1,size2);

   }
   }
```

```
#include<stdio.h>
#include<device.h>
#include<fmclient.h>
#include<gl.h>
#include <string.h>
#include "cpath.h"

#define sla  0.35
#define xla 0.325
#define yla  0.45

#define INFO 1
#define GO 2


 float compas[16][2] = {
{(xla +sla/2),yla},
{xla+(sla *3)/8,yla+sla/2},
{xla+sla/2,yla+sla},
{xla+(sla *5)/8,yla+sla/2},
·xla,yla+sla/2},
·xla+sla/2,yla+(sla *3)/8},
·xla+sla,yla+sla/2},
·xla+sla/2,yla+(sla * 5)/8},
·xla+(sla/4),yla+(sla * 3)/4},
·xla+(sla/4),yla+(sla/4)},
·xla+(sla *3)/4,yla+sla/4},
 xla+(sla *3)/4,yla+(sla*3)/4}
·;


 float whitevec2[] = {1.0,1.0,1.0};
'loat whitevec1[] = {0.5,0.8,1.0};
'loat backvec1[] = {0.0,1.0,0.0};
'loat bluevec[] = {0.0,0.0,1.0};
'loat yelvec[] = {1.0,1.0,0.0};
'loat redvec[] = {1.0,0.0,0.0};
 ong  xo,yo,xsize,ysize;
 ong igid, igid1;
 nt menu,menu1,menu2,submenu1,submenu2,submenu3,submenu4,submenu5;
.hort mx,my;
 nt mode,job,exitmode;
 har filename[30],runinfo[50],trianginfo[50],showinfo[50],leadersur[50];
 har surfaceinfo[50],intinfo[50],geomview[50],text[50];
   int max_size;


 main(argc,argv)
 int argc;
 char *argv[];
 {

  char *stroka[100],name[100] ;
  int i;

  FILE *fp1,*fp2,*fopen();
  if (argc< 3)
  printf("usage: geoplane [name of file] [number of points after interpolation]\n");
   else {
   sprintf(runinfo,"%s%s",PATH,"run.info");
   sprintf(trianginfo,"%s%s",PATH,"triang.info");
   sprintf(showinfo,"%s%s",PATH,"show.info");
   sprintf(intinfo,"%s%s",PATH,"int.info");
   sprintf(surfaceinfo,"%s%s",PATH,"surface.info");
   sprintf(geomview,"%s%s%s",PATH,"geomview",PLACE);
   sprintf(text,"%s%s",PATH,"text1");
   sprintf(leadersur,"%s%s",PATH,"leader.sur");
  max_size = atoi(argv[2]);
  fp1=fopen(leadersur,"r");
  fp2 = fopen("leader","w");
  i=1;
  while(fgets(stroka,100,fp1)!=NULL) {
   if(i == 20){
   if(max_size != 0)
   {
   fprintf(fp2,"%d\n",max_size);
   }
   else {
   sscanf(stroka,"%d",&max_size);
   fprintf(fp2,"%s",stroka);
   }
   }
   else{
   fprintf(fp2,"%s",stroka);
   }
   i++;
   }

   fclose(fp1);
   fclose(fp2);
  strcpy(filename,argv[1]);
  initialise();
  concave(TRUE);
  Drawobj();

  while(!exitmode){
  while(!qtest()){
  Drawobj();
  swapbuffers();
  }
  processinput();
       }
  }
  winclose(igid);
  system("clean");
  }

initialise()
{
job = -1;
exitmode = 0;
keepaspect(1,1);
foreground(0);
prefsize(700,700);
igid  = winopen("geoplane");
RGBmode():
doublebuffer();
gconfig();
ortho(0.0,1.0,0.0,1.0,-1.0,1.0);
c3f(backvec1);
clear();
qdevice(RIGHTMOUSE);


tie(RIGHTMOUSE,MOUSEX,MOUSEY);
getorigin(&xo,&yo);
getsize(&xsize,&ysize);
myMenus();
```

```
yMenus()


enu = defpup("GEOPLANE %t|INFO |GO ");
enu1 = defpup("EXIT GEOPLANE %t| YES | NO ");
/*
enu2 = newpup();
ddtopup(menu2,"OPTIONS %t");
ddtopup(menu2,"errors in the input file?%m",submenu1);
ddtopup(menu2,"size of grid %m",submenu2);
ddtopup(menu2,"radius of confidence%m",submenu3);
ddtopup(menu2,"smoothing %m ",submenu4);
ddtopup(menu2,"kill the peaks ? %m",submenu5);
 submenu1 = newpup();
ddtopup(submenu1," YES %f",adderr());
ddtopup(submenu1,"NO");
  submenu2 = newpup();

ddtopup("100 %f",gridsize(100));
ddtopup("1000 %f",gridsize(1000));
ddtopup("10000 %f",gridsize(10000));
   submenu3 = newpup();
ddtopup("100 %f",radofconf());
ddtopup("1000 %f",radofconf());
ddtopup("10000 %f",radofconf());

    submenu4 = newpup();
ddtopup("no smoothing %f", smooth(0));
ddtopup(" smoothing, according to errors %f", smooth(1));
ddtopup(" smooth (all errors = 1) %f", smooth(2));

    submenu5 = newpup();
ddtopup("no kill %f", smooth(0));
ddtopup(" kill, according to errors %f", kill(1));
ddtopup("kill by statistical methods %f", kill(2));
/




dderr()

rintf("adderr is working");



ridsize(i)
nt i;

rintf("gridsize is working %d \n,i");


adofconf(i)
nt i;

rintf("radofconf is working %d \n,i");

.ll(i)
nt i;
{
printf("kill is working %d \n,i");
}

smooth(i)
int i;
{
printf("smooth is working %d \n,i");

}


 Drawobj()
 {
 fmfonthandle f;
 fmfonthandle fsized;

 int i,k,ind,angle;
 float x0,y0,xre,yre,size,rad,rad1;
 char *str[15],stri[100],stri1[100],stri2[100];
 char string[50],st[100];
 str[0] ="lation";
 str[1] ="triangu-";
 str[2]= "geoplane";
 str[3]= "  run";
 str[4]="lation";
 str[5] ="interpo-";
 str[6]= "options";
 str[7] = "     " ;
 str[8] = "change";
 str[9] =" data ";
 str[10] ="the model";
 str[11] ="  view ";

size = 24.0;
rad  = 0.1;
rad1 = 0.09;
sprintf(stri," %s  %d  ",filename,max_size);
sprintf(stri1,"     GEOPLANE  ");

fminit();
f = fmfindfont("Times-Roman");
fsized = fmscalefont(f,size);
fmsetfont(fsized);

 c3f(backvec1);
 rectf(0.0,0.0,1.0,1.0);
 c3f(whitevec2);
 rectf(0.3 ,0.05,0.7,0.15);
 cmov2(0.35,0.095);
 c3f(bluevec);
 fmprstr((unsigned char * )stri);
 rectf(0.2,0.85,0.7,0.95);
 c3f(yelvec);
 cmov2(0.34,0.89);
 fmprstr((unsigned char *)stri1);


 x0 = 0.1;y0 = 0.2;
 ind = 0;
 for(i = 0;i<3;i++)  {
 for(k =0;k<2;k++)   {
 xre = x0+i*0.3+rad;
 yre = y0+k*0.4+rad;
```

```
 c3f(whitevec2);
 rectf(xre-rad,yre-rad,xre+rad,yre+rad);
 if(ind != job)
 c3f(bluevec);
 else c3f(redvec);
 rectf(xre-rad1,yre-rad1,xre+rad1,yre+rad1);
ind++;
 }
 }
 ind = 0;
 for(i = 0;i<3;i++)  {
 for(k =0; k< 2; k++)   {
 xre = x0+i*0.3;
 yre = y0+k*0.4;
 c3f(yelvec);
 cmov2(xre+0.02,yre+0.07);
 fmprstr((unsigned char * )str[ind]);
 cmov2(xre+0.03,yre+0.13);
 fmprstr((unsigned char * )str[ind+1]);
   ind = ind+2;
   }
   }
/*picture();*/
}



processinput()
{
short val;
int dev,i,k,m;
long menuval;
i = k = m = 0;



while(qtest() ) {
Drawobj();
switch(qread(&val)) {
case (REDRAW):
reshapeviewport();
getorigin(&xo,&yo);
getsize(&xsize,&ysize);
while('qtest() ) {
Drawobj();
swapbuffers();
}
break;
case (RIGHTMOUSE):
if(val){
qread(&mx);
qread(&my);
k++;
if(k == 1){
choicel();
if(job  == -2) {
menuval  = dopup(menu1);
 switch(menuval){
    case 1:
      exitmode = 1;
      break;
    case 2:
      exitmode = 0;
      break;
      default:
      break;
      }
      }

if(job > -1) {
menuval = dopup(menu);
   switch(menuval) {
       case INFO:
         mode = INFO;
Drawobj();
swapbuffers();
job = -1;
choice();
            break;
          case (GO) :
             mode = GO;
Drawobj();
swapbuffers();
choice();
job = -1;
            break;
            }
            }
            }
            }
default:
job = -1;
break;
}
}
}



choice()
{
 float xco,yco;
 int ind1,ind2,ind5;
xco = (float)(mx -xo )/(float)xsize;
yco =(float) (my -yo)/(float)ysize;
ind1 = 1;
ind2 = 0;
  if((xco<0.3)&&(xco>0.1)){
      if(yco<0.4&&yco>0.2)
      {
      if(mode == GO)
      {
      start_(filename,&ind2);
      trian_(&ind1,&ind5);
      job = -1;
     }
      if (mode == INFO)
     try(trianginfo);
job = -1;
     }
      else if (yco>0.6&&yco<0.8)
      {
      if(mode == GO)
      {
      ind1 = 0;
      start_(filename,&ind2);
      job = 0 ;
```

```
      Drawobj(); swapbuffers();
      trian_(&ind2,&ind5);
      if(ind5 == 1){
      Drawobj();
      swapbuffers();
      }
      else
      job = 2;
      Drawobj(); swapbuffers();
      int_ (&ind2);
      job = 5;
      Drawobj(); swapbuffers();
      system("rm tempor addual text");
      system (geomview);
      sleep(10);
     job = -1;
      }
      if (mode == INFO)
     try(runinfo);
     job = -1;
     }
      }


  else if ((xco<0.6)&&(xco>0.4)){
      if(yco<0.4&&yco>0.2)
      {
      if(mode == GO){
     int_(&ind1);
      system("rm  rescales tempor addual");
     job = -1;
     }
      if (mode == INFO){
      try(intinfo);
     job = -1;
     }
     }
     else if(yco>0.6&&yco<0.8)
     {
     system("jot leader &");
     job = -1;
     }
}



    else if((xco<0.9)&&(xco>0.7))
    if(yco>0.2&&yco<0.4){
    if(mode == GO){
     start_(filename,&ind2);
     show_(text);
     system("rm rescales");
     job = -1;
     }
     if(mode == INFO){
     try(showinfo);
     job = -1;
     }
     }
     else if(yco>0.6&&yco<0.8)
     {
     if(mode == GO){
     system("geomview<triang &");
     sleep(10);
      job = -1;
      }
      if(mode == INFO){
      try(surfaceinfo);
      job = -1;
      }
      }
      }


choicel()
{
 float xco,yco;
xco = (float)(mx -xo )/(float)xsize;
yco =(float) (my -yo)/(float)ysize;
  if((xco<0.3)&&(xco>0.1)){
      if(yco<0.4&&yco>0.2)
      job = 0;
      else if (yco>0.6&&yco<0.8)
      job = 1;
      }


  else if ((xco<0.6)&&(xco>0.4)){
      if(yco<0.4&&yco>0.2)
      job = 2;
      else if(yco>0.6&&yco<0.8)
      job = 3;
      }

  else if(xco<0.9&&xco>0.7){
      if(yco<0.4&&yco>0.2)
      job = 4;
      else if (yco>0.6&&yco<0.8)
      job = 5;
      }


      if(xco<0.7 && xco>0.3)
      {
      if(yco>0.85 && yco<0.95)
      job  = -2;
      }
      }




 picture()
 {



 concave(TRUE);
c3f(bluevec);
bgnpolygon();
v2f(&compas[8][0]);
v2f(&compas[1][0]);
```

```
v2f(&compas[9][0]);
v2f(&compas[5][0]);
v2f(&compas[10][0]);
v2f(&compas[3][0]);
v2f(&compas[11][0]);
v2f(&compas[7][0]);
v2f(&compas[8][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[8][0]);
v2f(&compas[1][0]);
v2f(&compas[9][0]);
v2f(&compas[5][0]);
v2f(&compas[10][0]);
v2f(&compas[3][0]);
v2f(&compas[11][0]);
v2f(&compas[7][0]);
v2f(&compas[8][0]);
endclosedline();
move2(compas[9][0],compas[9][1]);
draw2(compas[11][0],compas[11][1]);
move2(compas[8][0],compas[8][1]);
draw2(compas[10][0],compas[10][1]);
c3f(bluevec);
bgnpolygon();
v2f(&compas[4][0]);
v2f(&compas[5][0]);
v2f(&compas[6][0]);
v2f(&compas[7][0]);
v2f(&compas[4][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[4][0]);
v2f(&compas[5][0]);
v2f(&compas[6][0]);
v2f(&compas[7][0]);
v2f(&compas[4][0]);
endclosedline();
c3f(bluevec);
bgnpolygon();
v2f(&compas[0][0]);
v2f(&compas[1][0]);
v2f(&compas[2][0]);
v2f(&compas[3][0]);
v2f(&compas[0][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[0][0]);
v2f(&compas[1][0]);
v2f(&compas[2][0]);
v2f(&compas[3][0]);
v2f(&compas[0][0]);
endclosedline();


c3f(yelvec);
move2(compas[0][0],compas[0][1]);
draw2(compas[2][0],compas[2][1]);
move2(compas[4][0],compas[4][1]);
draw2(compas[1][0],compas[1][1]);
move2(compas[3][0],compas[3][1]);
draw2(compas[6][0],compas[6][1]);

}
```

```
#include<stdio.h>
#include<device.h>
#include<fmclient.h>
#include<gl.h>
#include <string.h>
#include "cpath.h"

#define sla  0.20
#define xla 0.005
#define yla  0.805

#define INFO 1
#define GO 2



 float compas[16][2] = {
{(xla +sla/2),yla},
{xla+(sla *3)/8,yla+sla/2},
{xla+sla/2,yla+sla},
{xla+(sla *5)/8,yla+sla/2},
{xla,yla+sla/2},
{xla+sla/2,yla+(sla *3)/8},
{xla+sla,yla+sla/2},
{xla+sla/2,yla+(sla * 5)/8},
{xla+(sla/4),yla+(sla * 3)/4},
{xla+(sla/4),yla+(sla/4)},
{xla+(sla *3)/4,yla+sla/4},
{xla+(sla *3)/4,yla+(sla*3)/4}
};


 float whitevec2[] = {1.0,1.0,1.0};
float whitevec1[] = {0.5,0.8,1.0};
float backvec1[] = {0.0,1.0,0.0};
float bluevec[] = {0.0,0.0,1.0};
float yelvec[] = {1.0,1.0,0.0};
float redvec[] = {1.0,0.0,0.0};
long  xo,yo,xsize,ysize;
long igid, igid1;
short mx,my;
int norm, peak,hardcopy;
float wlog;
char hardstr[100];
int mode,job,exitmode,getout;
char filename[50],runinfo[50],trianginfo[50],showinfo[50],lea ersur[50];
char surfaceinfo[50],intinfo[50],geomview[50],text[50],prepar[50],clean[50];
   int max_size;
  main(argc,argv)
  int argc;
  char *argv[];
  {

   char stroka[100],name[100] ;
   int i;

   FILE *fp1,*fp2, *fp, *fopen();
    sprintf(runinfo,"%s%s",PATH,"run.info");
    sprintf(trianginfo,"%s%s",PATH,"triang.info");
    sprintf(showinfo,"%s%s",PATH,"show.info");
    sprintf(intinfo,"%s%s",PATH,"int.info");
    sprintf(surfaceinfo,"%s%s",PATH,"surface.info");
    sprintf(geomview,"%s%s%s",PATH,"geomview",PLACE);
    sprintf(text,"%s%s",PATH,"text1");
    sprintf(leadersur,"%s%s",PATH,"leader.sur");
    sprintf(prepar,"%s%s",PATH,"prepar");
    sprintf(clean,"%s%s",PATH,"clean");
    system(clean);
   if (argc< 2) {
     system(prepar);
    fp = fopen("geoname","r");
             if(fp'= NULL) fscanf(fp,"%s",filename);}
    else strcpy(filename , argv[1]);
   fp1=fopen(leadersur,"r");
   fp2 = fopen("leader","w");
   while(fgets(stroka,100,fp1)!=NULL) {
    fprintf(fp2,"%s",stroka);
   }

    fclose(fp1);
    fclose(fp2);

  initialise();
  concave(TRUE);
  Drawobj();

  while(!exitmode){
  while(!qtest()){
  Drawobj();
  swapbuffers();
  }
  processinput();
        }
   winclose(igid);
  system(clean);
  }

initialise()
{
job = -1;
wlog = -1;
peak = -1;
norm = 0;
hardcopy = 0;
exitmode = 0;
keepaspect(1,1);
foreground();
prefsize(700,700);
igid  = winopen("geoplane");
RGBmode();
doublebuffer();
gconfig();
ortho(0.0,1.0,0.0,1.0,-1.0,1.0);
c3f(backvec1);
clear();
qdevice(RIGHTMOUSE);
tie(RIGHTMOUSE,MOUSEX,MOUSEY);
getorigin(&xo,&yo);
getsize(&xsize,&ysize);
myMenus();
}


int menu,menu1,menu2,menu3,menu4,menu5,menu6,menu7,menu8,menu9,menu10;
normalize(n)
int n;
{
norm = 1;
```

```
;


ogar(n)
nt n;

log = 8;


eaks(n)
nt n;

eak = n;


eset(n)
nt n;

log = -1;
eak = -1;
orm = 0;
ardcopy = 0;


et_max_size(n)
nt n;

ax_size = n;
eturn n;


estart(n)
nt n;

ILE *fp1,*fp2, *fp, *fopen();
har  stroka[100];
ystem(clean);
ystem(prepar);
    fp = fopen("geoname","r");
   if(fp!= NULL) fscanf(fp,"%s",filename);
 fp1=fopen(leadersur,"r");
  fp2 = fopen("leader","w");
  while(fgets(stroka,100,fp1)!=NULL) {
   fprintf(fp2,"%s",stroka);
   }
   fclose(fp1);
   fclose(fp2);
   fclose(fp);
rawobj();
wapbuffers();

ake_hardcopy()


ardcopy = 1;
printf(hardstr,"%s%s%s",
launch -h \"mv hardtmp\" -m \"Save hardcopy as\"  -c \"",
ilename,".mongo\"");



ave_triang()

char locstr[100];
sprintf(locstr,"%s%s%s","launch -h \"cp triang\" -m \"Save file as\" -c \"",
 filename,".tri\"");
system (locstr);
}


myMenus()
{

menu = defpup("GEOPLANE %t|INFO  |GO ");
menu1 = defpup("GEOPLANE %t");
addtopup(menu1," OPEN %f ",restart);
addtopup(menu1," SAVE %f| QUIT ",save_triang);
menu2 = defpup("Level of smoothing %t %F| 2 %x2| 4 %x4| 8 %x8",get_max_size);
menu7 = defpup("LOGARITHM %t %F| 2 %x2 |4 %x4| 8 %x8",logar);
menu8 = defpup("ERRORS IN THE FILE? %t %F |YES|NO",peaks);
menu9 = defpup("SMOOTH %t,|KILL PEAKS %m",menu8);
addtopup(menu9,"LOGARITHM %f",logar);
menu10 = defpup("OPTIONS %t |NORMALIZE %f",normalize);
addtopup(menu10,"SMOOTH %m",menu9);
addtopup(menu10,"HARDCOPY %f",make_hardcopy);
addtopup(menu10,"RESET ALL %f",reset);


}


 Drawobj()
 {
 fmfonthandle f;
 fmfonthandle fsized;

 int i,k,ind,angle,len;
 float x0,y0,xre,yre,size,rad,rad1;
 char *str1[15],str1[100],str11[100],str12[100];
 char string[50],st[100];
 str1[0] ="lation";
 str1[1] ="triangu-";
 str1[2]= "geoplane";
 str1[3]= "  run";
 str1[4]="lation";
 str1[5] ="interpo-";
 str1[6]= "options";
 str1[7] = "     " ;
 str1[8] = "change";
 str1[9] =" data ";
 str1[10] ="the model";
 str1[11] ="  view ";

size = 24.0;
rad  = 0.1;
rad1 = 0.09;
sprintf(str1,"  %s  ",filename);
sprintf(str11,"     GEOPLANE  ");

fminit();
f = fmfindfont("Times-Roman");
fsized = fmscalefont(f,size);
fmsetfont(fsized);

 c3f(backvec);
 rectf(0.0,0.0,1.0,1.0);
```

```
c3f(bluevec);
rectf(0.3,0.85,0.7,0.95);
 c3f(yelvec);
 cmov2(0.34,0.89);
 fmprstr(str11);



 x0 = 0.1;y0 = 0.2;
 ind = 0;
 for(i = 0;i<3;i++)  {
 for(k =0;k<2;k++)   {
 xre = x0+i*0.3+rad;
 yre = y0+k*0.4+rad;
 c3f(whitevec2);
 rectf(xre-rad,yre-rad,xre+rad,yre+rad);
 if(ind != job)
 c3f(bluevec);
 else c3f(redvec);
 rectf(xre-rad1,yre-rad1,xre+rad1,yre+rad1);
ind++;
 }
 }
 ind = 0;
 for(i = 0;i<3;i++)  {
 for(k =0; k< 2; k++)   {
 xre = x0+i*0.3;
 yre = y0+k*0.4;
 c3f(yelvec);
 cmov2(xre+0.02,yre+0.07);
 fmprstr(str1[ind]);
 cmov2(xre+0.03,yre+0.13);
 fmprstr(str1[ind+1]);
   ind = ind+2;
   }
   }

  len = (strlen(filename) + 4)*0.7;
  size = (0.4*xsize)/(float)len;
  if(size>24)size = 24;
  fsized = fmscalefont(f,size);
  fmsetfont(fsized);

 c3f(whitevec2);
 rectf(0.3 ,0.05,0.7,0.15);
 cmov2(0.35,0.035);
 c3f(bluevec);
 fmprstr(str1);

picture();
}



processinput()
{
short val;
int dev,i,k,m;
long menuval;
i = k = m = 0;



while(qtest() ) {
Drawobj();
switch(qread(&val)) {
case (REDRAW):
reshapeviewport();
getorigin(&xo,&yo);
getsize(&xsize,&ysize);
while(!qtest() ) {
Drawobj();
swapbuffers();
}
break;
case (RIGHTMOUSE):
if(val){
qread(&mx);
qread(&my);
k++;
if(k == 1){
choice1();
if(job  == -2) {
menuval  = dopup(menu1);
 switch(menuval){
    case 3:
      exitmode = 1;
      break;
       default:
      break;
      }
      }

if(job == 3){
 menuval = dopup(menu10);
 Drawobj();
swapbuffers();
job = -1;
choice();
}
if(job > -1&&job!=3) {
menuval = dopup(menu);
   switch(menuval) {
       case INFO:
         mode = INFO;
Drawobj();
swapbuffers();
job = -1;
choice();
            break;
         case (GO) :
            mode = GO;
Drawobj();
swapbuffers();
choice();
job = -1;
            break;
            }
            }
            }
            }
default:
job = -1;
break;
}
}
}
```

```
choice()
{
 float xco,yco;
 int ind1,ind2,ind5;
 long menuval1;
xco = (float)(mx -xo )/(float)xsize;
yco =(float) (my -yo)/(float)ysize;
ind1 = 1;
ind2 = 0;
  if((xco<0.3)&&(xco>0.1)){
      if(yco<0.4&&yco>0.2)
      {
      if(mode == GO)
      {
      start_(filename,&norm,&wlog,&peak);
      trian_(&hardcopy);
      if(hardcopy)system(hardstr);
      job = -1;
     }
      if (mode == INFO)
     try(trianginfo);
job = -1;
     }
      else if (yco>0.6&&yco<0.8)
      {
       if(mode == GO){
         ind1 = 0;
         max_size = dopup(menu2);
        start_(filename,&norm,&wlog,&peak);
         job = 0 ;
         Drawobj(); swapbuffers();
        trian_(&hardcopy);
         if(max_size > 1){
          job = 2;
           Drawobj(); swapbuffers();
          add_(&max_size);
          int_ (&max_size);
           if(hardcopy)system(hardstr);
          }
         job = 5;
         Drawobj(); swapbuffers();
         system("rm tempor addual text");
        system (geomview);
         sleep(10);
         job = -1;
       }
      if (mode == INFO)
     try(runinfo);
     job = -1;
     }
      }


  else if ((xco<0.6)&&(xco>0.4)){
      if(yco<0.4&&yco>0.2)
      {
      if(mode == GO){
       max_size = dopup(menu2);
        if(max_size>1){
         add_(&max_size);
         Drawobj(); swapbuffers();
         int_(&max_size);
         system("rm  rescales tempor addual");
         job = -1;
        }
      }
         if (mode == INFO){
          try(intinfo);
          job = -1;
         }
      }

}



     else if((xco<0.9)&&(xco>0.7))
     if(yco>0.2&&yco<0.4){
     if(mode == GO){
      start_(filename,&norm,&wlog,&peak);
      show_(text,&hardcopy);
      if(hardcopy)system(hardstr);
      system("rm rescales");
      job = -1;
      }
      if(mode == INFO){
      try(showinfo);
      job = -1;
      }
      }
      else if(yco>0.6&&yco<0.8)
      {
      if(mode == GO){
      system(geomview);
      sleep(10);
      job = -1;
      }
      if(mode == INFO){
      try(surfaceinfo);
      job = -1;
      }
      }
      }



choicel()
{
 float xco,yco;
xco = (float)(mx -xo )/(float)xsize;
yco =(float) (my -yo)/(float)ysize;
  if((xco<0.3)&&(xco>0.1)){
      if(yco<0.4&&yco>0.2)
      job = 0;
      else if (yco>0.6&&yco<0.8)
      job = 1;
      }


  else if ((xco<0.6)&&(xco>0.4)){
      if(yco<0.4&&yco>0.2)
      job = 2;
      else if(yco>0.6&&yco<0.8)
      job = 3;
      }
```

```
  else if(xco<0.9&&xco>0.7){
      if(yco<0.4&&yco>0.2)
      job = 4;
      else if (yco>0.6&&yco<0.8)
      job = 5;
      }


      if(xco<0.7 && xco>0.3)
      {
      if(yco>0.85 && yco<0.95)
      job  = -2;
      }
      }




picture()
{



 concave(TRUE);
·3f(bluevec);
)gnpolygon();
·2f(&compas[8][0]);
·2f(&compas[1][0]);
·2f(&compas[9][0]);
·2f(&compas[5][0]);
·2f(&compas[10][0]);
·2f(&compas[3][0]);
·2f(&compas[11][0]);
·2f(&compas[7][0]);
·2f(&compas[8][0]);
·ndpolygon();
 3f(yelvec);
)gnclosedline();
 2f(&compas[8][0]);
 2f(&compas[1][0]);
 2f(&compas[9][0]);
 2f(&compas[5][0]);
 2f(&compas[10][0]);
 2f(&compas[3][0]);
 2f(&compas[11][0]);
 2f(&compas[7][0]);
 2f(&compas[8][0]);
ndclosedline();
ove2(compas[9][0],compas[9][1]);
raw2(compas[11][0],compas[11][1]);
ove2(compas[8][0],compas[8][1]);
raw2(compas[10][0],compas[10][1]);
3f(bluevec);
gnpolygon();
2f(&compas[4][0]);
2f(&compas[5][0]);
2f(&compas[6][0]);
2f(&compas[7][0]);
v2f(&compas[4][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[4][0]);
v2f(&compas[5][0]);
v2f(&compas[6][0]);
v2f(&compas[7][0]);
v2f(&compas[4][0]);
endclosedline();
c3f(bluevec);
bgnpolygon();
v2f(&compas[0][0]);
v2f(&compas[1][0]);
v2f(&compas[2][0]);
v2f(&compas[3][0]);
v2f(&compas[0][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[0][0]);
v2f(&compas[1][0]);
v2f(&compas[2][0]);
v2f(&compas[3][0]);
v2f(&compas[0][0]);
endclosedline();


c3f(yelvec);
move2(compas[0][0],compas[0][1]);
draw2(compas[2][0],compas[2][1]);
move2(compas[4][0],compas[4][1]);
draw2(compas[1][0],compas[1][1]);
move2(compas[3][0],compas[3][1]);
draw2(compas[6][0],compas[6][1]);

}
```

geoform2.c

```
#include<stdio.h>
#include<device.h>
#include<fmclient.h>
#include<gl.h>
#include <string.h>
#include "cpath.h"

#define sla  0.35
#define xla 0.325
#define yla  0.45

#define INFO 1
#define GO 2


float compas[16][2] = {
{(xla +sla/2),yla},
{xla+(sla *3)/8,yla+sla/2},
{xla+sla/2,yla+sla},
{xla+(sla *5)/8,yla+sla/2},
{xla,yla+sla/2},
{xla+sla/2,yla+(sla *3)/8},
{xla+sla,yla+sla/2},
{xla+sla/2,yla+(sla * 5)/8},
{xla+(sla/4),yla+(sla * 3)/4},
{xla+(sla/4),yla+(sla/4)},
{xla+(sla *3)/4,yla+sla/4},
{xla+(sla *3)/4,yla+(sla*3)/4}
};


 float whitevec2[] = {1.0,1.0,1.0};
float whitevec1[] = {0.5,0.8,1.0};
float backvec1[] = {1.0,1.0,0.0};
float bluevec[] = {0.0,0.0,1.0};
float yelvec[] = {1.0,1.0,1.0};
float redvec[] = {1.0,0.0,0.0};
long  xo,yo,xsize,ysize;
long igid, igid1;
int menu,menu1;
int ind5;
short mx,my;
int mode,job,exitmode;
char filename[30],sruninfo[50],addsinfo[50],showinfo[50];
char surfaceinfo[50],intsinfo[50],sgeomview[50];
   int max_size;


  main(argc,argv)
  int argc;
  char *argv[];
  {

   char *stroka[100],name[100] ;
   int i;

   FILE *fp1,*fp2,*fopen();
   if (argc< 3)
   printf("usage: geosphere [name of file] [number of points after interpolation]\n");
    else {
    sprintf(sruninfo,"%s%s",PATH,"srun.info");
    sprintf(addsinfo,"%s%s",PATH,"adds.info");
    sprintf(showinfo,"%s%s",PATH,"show.info");
    sprintf(intsinfo,"%s%s",PATH,"ints.info");
    sprintf(surfaceinfo,"%s%s",PATH,"surface.info");
    sprintf(sgeomview,"%s%s%s",PATH,"sgeomview",PLACE),
   max_size = atoi(argv[2]);
  strcpy(filename,argv[1]);
  initialise();
  concave(TRUE);
  Drawobj();

  while('exitmode){
  while(!qtest()){
  Drawobj();
  swapbuffers();
  }
  processinput();
        }
  }
  winclose(igid);
  }

initialise()
{
job = -1;
exitmode = 0;
keepaspect(1,1);
foreground(0);
prefsize(700,700);
igid  = winopen("geosphere");
RGBmode();
doublebuffer();
gconfig();
ortho(0.0,1.0,0.0,1.0,-1.0,1.0);
c3f(backvec1);
clear();
qdevice(RIGHTMOUSE);
menu = defpup("GEOSPHERE %t|INFO  |GO ");
menu1 = defpup("EXIT GEOSPHERE %t| YES | NO ");
tie(RIGHTMOUSE,MOUSEX,MOUSEY);
getorigin(&xo,&yo);
getsize(&xsize,&ysize);
}


 Drawobj()
 {
 fmfonthandle f;
 fmfonthandle fsized;

 int i,k,ind,angle;
 float x0,y0,xre,yre,size,rad,rad1;
 char *str[15],stri[100],stri1[100],stri2[100];
 char string[50],st[100];
 str[0] ="lation";
 str[1] ="triangu-";
 str[2]= "geosphere";
 str[3]= "  run";
 str[4]="lation";
 str[5] ="interpo-";
 str[6] = "change";
 str[7] =" data ";
 str[8] ="the model";
 str[9] ="  view ";

size = 18.0;
rad  = 0.1;
```

```
radl = 0.09;
sprintf(stri," %s  %d  ",filename,max_size);
sprintf(stri1,"     GEOSPHERE  ");

fminit();
f = fmfindfont("Times-Roman");
fsized = fmscalefont(f,size);
fmsetfont(fsized);

 c3f(backvec1);
 rectf(0.0,0.0,1.0,1.0);
 c3f(whitevec2);
 rectf(0.3 ,0.05,0.7,0.15);
 cmov2(0.35,0.095);
 c3f(bluevec);
 fmprstr((unsigned char * )stri);
 rectf(0.3,0.85,0.7,0.95);
 c3f(yelvec);
 cmov2(0.34,0.89);
 fmprstr((unsigned char *)stri1);


 x0 = 0.1;y0 = 0.2;
 ind = 0;
 for(i = 0;i<3;i++)  {
 for(k =0; k< 2; k++)   {
 if(i * k != 1) {
 xre = x0+i*0.3+rad;
 yre = y0+k*0.4+rad;
 c3f(whitevec2);
 rectf(xre-rad,yre-rad,xre+rad,yre+rad);
 if(ind != job)
 c3f(bluevec);
 else c3f(redvec);
 rectf(xre-radl,yre-radl,xre+radl,yre+radl);
 ind++;
 }

 }
 }
 ind = 0;
 for(i = 0;i<3;i++)  {
 for(k =0; k< 2; k++)   {
 if(i*k!=1) {
 xre = x0+i*0.3;
 yre = y0+k*0.4;
 c3f(yelvec);
 cmov2(xre+0.02,yre+0.07);
 fmprstr((unsigned char * )str[ind]);
 cmov2(xre+0.03,yre+0.13);
 fmprstr((unsigned char * )str[ind+1]);
   ind = ind+2;
   }
   }

 ucture();



 rocessinput()

 nort val;
 nt dev,i,k,m;

long menuval;
i = k = m = 0;


while(qtest() ) {
Drawobj();
switch(qread(&val)) {
case (REDRAW):
reshapeviewport();
getorigin(&xo,&yo);
getsize(&xsize,&ysize);
while(!qtest() ) {
Drawobj();
swapbuffers();
}
break;
case (RIGHTMOUSE):
if(val){
qread(&mx);
qread(&my);
k++;
if(k == 1){
choice1();
if(job  == -2) {
menuval  = dopup(menu1);
 switch(menuval){
    case 1:
      exitmode = 1;
      break;
    case 2:
      exitmode = 0;
      break;
      default:
      break;
      }
      }

if(job > -1) {
menuval = dopup(menu);
   switch(menuval) {
       case INFO:
         mode = INFO;
Drawobj();
swapbuffers();
job = -1;
choice();
            break;
         case (GO) :
            mode = GO;
Drawobj();
swapbuffers();
choice();
job = -1;
            break;
            }
            }
            }
            }
default:
job = -1;
break;
}
}
```

```
}



choice()
{
 float xco,yco;
 int ind1,ind2,ind6,ind7;
xco = (float)(mx -xo )/(float)xsize;
yco =(float) (my -yo)/(float)ysize;
ind1 = 1;
ind2 = 0;
ind6 = 0;
ind7 = 1;
  if((xco<0.3)&&(xco>0.1)){
      if(yco<0.4&&yco>0.2)
      {
      if(mode == GO)
      {
      adds_(filename,&max_size,&ind5,&ind6);
     job = -1;
     }
      if (mode == INFO)
     try(addsinfo);
     job = -1;
     }
      else if (yco>0.6&&yco<0.8)
      {
      if(mode == GO)
      {

      sleep(2);
      job = 0;
      Drawobj();swapbuffers();
      adds_(filename,&max_size,&ind5,&ind6);
      if(ind5 == 0) {
      job = 2;
      Drawobj();swapbuffers();
      ints_(&ind5,&ind7);

      }
      else
      {
      job = 2;
      Drawobj(); swapbuffers();
      ints_ (&ind5,&ind6);
     }
      job = 4;
      Drawobj(); swapbuffers();
      system("rm saddual stempor");
      system (sgeomview);
      sleep(10);
     job = -1;
      }
      if (mode == INFO)
     try(sruninfo);
     job = -1;
     }
      }


 else if ((xco<0.6)&&(xco>0.4)){
     if(yco<0.4&&yco>0.2)
      {
      if(mode == GO){
      ints_(&ind5,&ind7);
      system("rm saddual stempor");
     job = -1;
     }
      if (mode == INFO){
      try(intsinfo);
     job = -1;
     }
     }
     }



     else if((xco<0.9)&&(xco>0.7))
     if(yco>0.2&&yco<0.4){
     if(mode == GO){
       job = -1;
          adds_(filename,&max_size,&ind5,&ind7);
      }
      if(mode == INFO){
      try(showinfo);
      job = -1;
      }
      }
      else if(yco>0.6&&yco<0.8)
      {

if(mode == GO){
      system("sgeomview<triang &");
      sleep(10);
      job = -1;
      }
      if(mode == INFO){
      try(surfaceinfo);
      job = -1;
      }
      }
      }



choicel()
{
 float xco,yco;
xco = (float)(mx -xo )/(float)xsize;
yco =(float) (my -yo)/(float)ysize;
  if((xco<0.3)&&(xco>0.1)){
      if(yco<0.4&&yco>0.2)
      job = 0;
      else if (yco>0.6&&yco<0.8)
      job = 1;
      }


  else if ((xco<0.6)&&(xco>0.4)){
      if(yco<0.4&&yco>0.2)
      job = 2;
      }

  else if(xco<0.9&&xco>0.7){
      if(yco<0.4&&yco>0.2)
      job = 3;
```

```
      else if (yco>0.6&&yco<0.8)
      job = 4;
      }


      if(xco<0.7 && xco>0.3)
      {
      if(yco>0.85 && yco<0.95)
      job  = -2;
      }
      }




 picture()
 {



  concave(TRUE);
c3f(bluevec);
bgnpolygon();
v2f(&compas[8][0]);
v2f(&compas[1][0]);
v2f(&compas[9][0]);
v2f(&compas[5][0]);
v2f(&compas[10][0]);
v2f(&compas[3][0]);
v2f(&compas[11][0]);
v2f(&compas[7][0]);
v2f(&compas[8][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[8][0]);
v2f(&compas[1][0]);
v2f(&compas[9][0]);
v2f(&compas[5][0]);
v2f(&compas[10][0]);
v2f(&compas[3][0]);
v2f(&compas[11][0]);
v2f(&compas[7][0]);
v2f(&compas[8][0]);
endclosedline();
move2(compas[9][0],compas[9][1]);
draw2(compas[11][0],compas[11][1]);
move2(compas[8][0],compas[8][1]);
draw2(compas[10][0],compas[10][1]);
c3f(bluevec);
bgnpolygon();
v2f(&compas[4][0]);
v2f(&compas[5][0]);
v2f(&compas[6][0]);
v2f(&compas[7][0]);
v2f(&compas[4][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[4][0]);
v2f(&compas[5][0]);
v2f(&compas[6][0]);
v2f(&compas[7][0]);
v2f(&compas[4][0]);
endclosedline();
c3f(bluevec);
bgnpolygon();
v2f(&compas[0][0]);
v2f(&compas[1][0]);
v2f(&compas[2][0]);
v2f(&compas[3][0]);
v2f(&compas[0][0]);
endpolygon();
c3f(yelvec);
bgnclosedline();
v2f(&compas[0][0]);
v2f(&compas[1][0]);
v2f(&compas[2][0]);
v2f(&compas[3][0]);
v2f(&compas[0][0]);
endclosedline();


c3f(yelvec);
move2(compas[0][0],compas[0][1]);
draw2(compas[2][0],compas[2][1]);
move2(compas[4][0],compas[4][1]);
draw2(compas[1][0],compas[1][1]);
move2(compas[3][0],compas[3][1]);
draw2(compas[6][0],compas[6][1]);

}
```

```
#include <stdio.h>
#include <math.h>
#include <gl.h>
#include "panel.h"
#include <device.h>
#include<string.h>
#include "cpath.h"
#define  maxcol  256
#define   k     3.5
#define   ANY    -1
#define   SIZE   50000


float  color_palette[maxcol][3];
int ncol;
Object Vobj,sphere,Vobj1;
float wx2,wy2,wz2,scal_par;
typedef float   Vec[3] ;

typedef struct {
        int     p_nsides ;
        Vec     ** p ;
        int      *s ;
} Polygon ;

 int    negflag ;
/*float         backvec[] = {0.8, 0.8, 0.8} ;*/
float   backvec[] = {0., 0., 0.} ;
float   blackvec[] = {0.0, 0.0, 0.0} ;
float   redvec[] = {1.0, 0.0, 0.0} ;
float   whitevec[] = {1.0, 1.0, 1.0} ;
float   yelvec[] = {1.0,1.0,0.0};


float idmat[] = {1.0, 0.0, 0.0, 0.0,
                 0.0, 1.0, 0.0, 0.0,
                 0.0, 0.0, 1.0, 0.0,
                 0.0, 0.0, 0.0, 1.0} ;
float OldMatrix[16];

Polygon *
makePolygon(nsides)
 int nsides ;

        Polygon * p ;

        p = (Polygon *) malloc (sizeof(Polygon)) ;

        p -> p_nsides = nsides ;

        p -> p = (Vec **) calloc(nsides, sizeof(Vec *)) ;
        p -> s = (int * ) malloc(3* sizeof(int )) ;

        return p ;


Vec     V[SIZE] ;
Polygon * P[SIZE] ;
Vec     N[SIZE],N1[SIZE] ;
int     npoints, npolys ;
Vec     mins = {1000000, 1000000, 1000000} ;
Vec     maxs = {-1000000, -1000000, -1000000} ;
float   V1[SIZE] ,V2[SIZE][3];

#define min(a, b) ((a) < (b) ? (a) : (b))
#define max(a, b) ((a) > (b) ? (a) : (b))

float Max ,scal;

#define         MKHSLIDER(v, n, b, t, i)\
        v = pnl_mkact(pnl_vslider) ;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v->w = 0.6;\
        v->h = 4.0;\
        v -> x = x ; v -> y = y ;\
        pnl_addact(v, p)


#define         LABEL(v,n)\
        v = pnl_mkact(pnl_label);\
        v->label = n;\
        v->x = x;\
        v->y = y;\
         pnl_addact(v,p)

#define         BUTTON(v, n, b, t, i)\
        v = pnl_mkact(pnl_radio_button) ;\
        v->labeltype = PNL_LABEL_BOTTOM;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = h1;v -> w = w;\
        pnl_addact(v, p)
#define         BUTTON2(v, n, b, t, i)\
        v = pnl_mkact(pnl_toggle_button) ;\
        v->labeltype = PNL_LABEL_BOTTOM;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = h1;v -> w = w;\
        pnl_addact(v, p)

#define         BUTTON3(v, n, b, t, i)\
        v = pnl_mkact(pnl_button) ;\
        v->labeltype = PNL_LABEL_BOTTOM;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = h1;v -> w = w;\
        pnl_addact(v, p)

Actuator * editPalette,* distance, * theta, * phi ,* model,*xlight1,*ylight1,
*zlight1,*xlight2,*ylight2,*zlight2,*xlight3,*ylight3,*zlight3,*surface,
*zsc, *map,*twist,*size, *shix,*shiy,*smooth,*xsc,*ysc,*tumble, *savePosition,
*getPosition, *myMouse, *savePict, *inputStr,*box, *reset,*quit, *l1,*l2,*l3,
*light1,*light2,*light3;

DefPanel()
{
```

```
Panel   * p ;
float   x = 0.0, y = 0.0, sep = 0.6,sep1=1.1 ,sep2 = 0.4;
float   hl= 0.5, w = 1.0;

p = pnl_mkpanel() ;
p -> label = "Control Panel v.10" ;
p -> ppu = 36.0 ;
BUTTON3(quit,"QUIT",0.0,1.0,0.0);
x = 4.5;y = quit->h+sep;
BUTTON3(savePict,"SAVE PICTURE",0.0,1.0,0.0);
y+= savePict->h+sep;
inputStr = pnl_mkact(pnl_typein);
inputStr->labeltype = PNL_LABEL_TOP_LEFT;
inputStr ->label = "Save as: ";
inputStr->x = 0.0;inputStr->y = y;
pnl_addact(inputStr,p);
y += inputStr->h+2*sep;
x = 2.0;
BUTTON2(box,"SHOW BOX",0.0,1.0,0.0);
x = 5.0;
BUTTON3(reset,"RESET",0.0,1.0,0.0);
y +=reset->h+2*sep;
x = 1.0;
BUTTON3(savePosition,"Save Position",0.0,1.0,0.0);
x = 4.5;
BUTTON3(getPosition,"Get Position", 0.0,1.0,0.0);
x = 8.0;
BUTTON3(editPalette,"Edit Palette",0.0,1.0,0.0);
x = 0.5;y+ = w+sep;
LABEL(12,"ROTATE");
x = 3.0;
LABEL(11,"TRANSLATE");
x = 6.0;
LABEL(13,"SCALE");
y+=hl+sep;x = 0.0;
MKHSLIDER(phi, "X ", 0.0, 3600.0, 0.0) ;
x+=phi->w+0.3;
MKHSLIDER(twist, "Y ", 0.0, 3600.0, 0.0) ;
x+=phi->w+0.3;
MKHSLIDER(theta, "Z ", 0.0, 3600.0, 0.0) ;
x+=phi->w+0.6;
MKHSLIDER(shix, " X", -1.0*Max, Max, 0.0) ;
x+=theta->w+0.3;
MKHSLIDER(shiy, " Y", -1.0*Max, Max, 0.0) ;
 x+=theta->w+0.3;
MKHSLIDER(distance, " Z", -2.0,2.0,0.0) ;
x+=theta->w+0.6;
MKHSLIDER(xsc, "X ",.1,10, 1.0) ;
x+=theta->w+0.3;
MKHSLIDER(ysc, "Y ",.1,10 ,1.0) ;
x+=theta->w+0.3;
MKHSLIDER(zsc, "Z ",.1,10 ,1.0) ;
x+=theta->w+0.6;
MKHSLIDER(size, "SIZE", 0.5,5.0 ,2.0/Max) ;
y+=size->h +1.5*sep;
x = 0.5;
BUTTON2(light1,"LIGHT",0.0,1.0,1.0);
x = 3.5;
BUTTON2(light2,"RED LIGHT",0.0,1.0,0.0);
x = 6.5;
BUTTON2(light3,"BLUE LIGHT",0.0,1.0,0.0);
y+=hl+sep;
x = 0.0;
MKHSLIDER(xlight1,"X",-1.0,1.0,0.0);
x+=theta->w+0.3;
MKHSLIDER(ylight1,"Y",-1.0,1.0,0.0);
x+=theta->w+0.3;
MKHSLIDER(zlight1,"Z",-1.0,1.0,1.0);
x+=theta->w+0.6;
MKHSLIDER(xlight2,"X",-1.0,1.0,0.0);
x+=theta->w+0.3;
MKHSLIDER(ylight2,"Y",-1.0.1.0,0.0);
x+=theta->w+0.3;
MKHSLIDER(zlight2,"Z",-1.0,1.0,1.0);
x+=theta->w+0.6;
MKHSLIDER(xlight3,"X",-1.0,1.0,0.0);
x+=theta->w+0.3;
MKHSLIDER(ylight3,"Y",-1.0,1.0,0.0);
x+=theta->w+0.3;
MKHSLIDER(zlight3,"Z",-1.0,1.0,1.0);
y+=zlight2->h+1.5*sep;
x = 2.0;
BUTTON (smooth,"Gouraud Lighting",0.0,1.0,0.0);
x = 6.0;
BUTTON (map,"3D Map",0.0,1.0,0.0);
y +=hl+sep;x = 4.0;
BUTTON2 (tumble,"Tumble",0.0,1.0,0.0);
y+=hl+sep;x = 2.0;
BUTTON(surface, "Lighting", 0.0, 1.0, 0.0) ;
x = 6.0;
BUTTON(model, "Wireframe",  0.0, 1.0, 0.0) ;
myMouse = pnl_mkact(pnl_mouse);
pnl_addact(myMouse,p);


}
        float shiftx ,shifty,tempo ;

ReadObject()
{

        int i, j, ku,nverts, index[3], nsides ;
        Vec     tmp1, tmp2 ;

        scanf("%d %d\n", & npolys, & npoints) ;

/*
        fprintf(stderr, "%d points, %d polys\n", npoints, npolys) ;
        */
                for (j = 0 ; j < 3 ; j++) {
                        mins[j] =  1e10;
                        maxs[j] =   -1e10;
                }

        for (i = 0 ; i < npoints ; i++) {
                scanf("%f %f %f\n", &(V[i][0]), &(V[i][1]), &(V[i][2])) ;
        }

/*Lala: I changed here */
        for(i = 0;i<npoints;i++)
                for (ku = 0 ; ku < 3 ; ku++) {
                        mins[ku] = min(mins[ku], V[i][ku]) ;
                        maxs[ku] = max(maxs[ku], V[i][ku]) ;
                    }
        Max = max(maxs[0]-mins[0], maxs[1]-mins[1]) ;
        Max = max(Max, maxs[2]-mins[2]) ;
        shiftx = 0.5*(maxs[0]+mins[0]);
        shifty = 0.5*(maxs[1]+mins[1]);
```

```
        for (i = 0; i < npoints ; i++) {
        tempo=V[i][1];
        V[i][0] = V[i][0]- shiftx;
        V[i][1] = tempo - shifty;
        }

        printf("shifts %f %f \n",shiftx,shifty);

        for (i = 0 ; i < npolys ; i++) {

                P[i] = MakePolygon(3) ;
                for (j = 0 ; j < 3 ; j++) {
                        scanf("%d", &index[j]) ;
                        P[i]->s[j] = index[j] -1 ;
                        P[i]->p[j] = V + (index[j] - 1) ;
                        V2[i][j] = V[index[j] -1][2];
                for (ku = 0 ; ku < 3 ; ku++) {
                        mins[ku] = min(mins[ku], V[index[j]-1][ku]) ;
                        maxs[ku] = max(maxs[ku], V[index[j]-1][ku]) ;
                }
                }
           V1[i] = V2[i][0] + V2[i][1] +V2[i][2];
           V1[i] = V1[i]/3.;
        }


        fprintf(stderr, "bbx: %f - %f\n", mins[0], maxs[0]) ;
        fprintf(stderr, "bby: %f - %f\n", mins[1], maxs[1]) ;
        fprintf(stderr, "bbz: %f - %f\n", mins[2], maxs[2]) ;
        */
                /*mins[2]=0;*/
        scal=ncol/(maxs[2]-mins[2]);

        Max = max(maxs[0]-mins[0], maxs[1]-mins[1]) ;
        Max = max(Max, maxs[2]-mins[2]) ;
        shiftx = 0.5*(maxs[0]+mins[0]);
        shifty = 0.5*(maxs[1]+mins[1]);
        scal_par = Max/2.0;
        printf("shifts %f %f \n",shiftx,shifty);

        for (i = 0 ; i < npolys ; i++) {
                VecSub(P[i]->p[0], P[i]->p[1], tmp1) ;
                VecSub(P[i]->p[2], P[i]->p[1], tmp2) ;
                VecCross(tmp1, tmp2, N[i]) ;
                VecNormalize(N[i]) ;
                if (negflag) {
                  if(N[i][0] < 0.)N[i][0] = - N[i][0];
                  if(N[i][1] < 0.)N[i][1] = - N[i][1];
                  if(N[i][2] < 0.)N[i][2] = - N[i][2];

                        }

             else{

                N[i][0] = - N[i][0] ;
                        N[i][1] = - N[i][1] ;
                        N[i][2] = - N[i][2] ;
                }
        }
        for (i = 0; i < npolys ; i++) {
        N1[i][0] = 0.0;
        N1[i][1] = 0.0;
        N1[i][2] = 0.0;
                        }

                for (i = 0; i < npolys ; i++) {
                for (j = 0; j< 3;j++) {
                N1[ P[i]->s[j] ][0]+= N[i][0];
                N1[ P[i]->s[j] ][1]+= N[i][1];
                N1[ P[i]->s[j] ][2]+= N[i][2];
                }
                  }
                for (i = 0; i < npolys ; i++)
                 VecNormalize(N1[i]);



        }

        VecNegate(a)
         Vec a ;
        {
                a[0] = 0 - a[0] ;
                a[1] = 0 - a[1] ;
                a[2] = 0 - a[2] ;
        }

        VecSub(a, b, c)
         Vec a, b, c ;
        {
                c[0] = a[0] - b[0] ;
                c[1] = a[1] - b[1] ;
                c[2] = a[2] - b[2] ;
        }

        VecCross(a, b, c)
         Vec a, b, c ;
        {
                c[0] = a[1] * b[2] - b[1] * a[2] ;
                c[1] = - (a[0] * b[2] - b[0] * a[2]) ;
                c[2] = a[0] * b[1] - b[0] * a[1] ;
        }

        VecNormalize(a)
         Vec a ;
        {
                float l ;
                l = (float) sqrt(a[0] * a[0] + a[1] * a[1] + a[2] * a[2]) ;

                a[0] /= l ;
                a[1] /= l ;
                a[2] /= l ;
        }

        float shinymaterial[] = { SPECULAR, 1.0, 0.2, 0.8,
                                  EMISSION , 0.8,0.8,0.0,
                                    DIFFUSE, 0.8, 0.0, 0.0,
                                    AMBIENT, 0.2, 0.0, 0.0,
                                    SHININESS, 3.0,
                                    LMNULL } ;
        float sun1[] = { LCOLOR,1.0, 1.0,1.0,
                         POSITION, 0.0, 0.0, 1.0,0.0,
                         LMNULL } ;
```

```
float sun2[] = { LCOLOR,1.0, 0 0,0.0,
                 POSITION, 0.0, 0.0, 1.0,0.0,
                 LMNULL } ;

float sun3[] = { LCOLOR,0.0, 0.0,1.0,
                 POSITION, 0.0, 0.0, 1.0,0.0,
                 LMNULL } ;

int tum ,tumbl;

DrawObject()
{
        Mouse * mymouse;
        short xmouse,ymouse;
        char * str[100];
        int i, j,col_ind ;
        int temp0,temp1,temp2,temp3,tumbl,editPal,getPos,savePos;
        float siz,shx,shy,x_sc,y_sc,z_sc;
        temp1=model -> val;
        temp2=surface -> val;
        temp0=smooth -> val;
        temp3=map -> val;
        siz = size -> val;
        shx = shix ->val;
        shy = shiy ->val;
        x_sc = xsc->val;
        y_sc = ysc->val;
        z_sc = zsc->val;
        editPal = editPalette ->val;
        getPos = getPosition -> val;
        savePos = savePosition -> val;

        xmouse = PNL_ACCESS(Mouse, myMouse, x);
        ymouse = PNL_ACCESS(Mouse, myMouse, y);
        /*get_palette();*/get_surface();
       sprintf(str,"%s%s %d %s",PATH,"mycedit",ncol,"&");
        if(editPal == 1.0) system(str);
        if(savePos == 1.0)save_position();
        if(getPos == 1.0) get_position();
        if(savePict->val)save_picture();
        if(reset->val)make_reset1();
        reshapeviewport();
        c3f(backvec) ;
        clear() ;
        zclear() ;
        MakeVobj();
        callobj(Vobj1);
        /*MakeLight(xmouse,ymouse);*/
        MakeLight1();
    /*   callobj(Vobj);*/
        if(box->val)Drawbox();
        /*translate(-1.0*shiftx,-1.0*shifty,0.0);*/
        ChangeStack();
        if (temp1 == 1)  {
lmbind(MATERIAL, 0) ;
lmbind(LIGHT0, 0) ;
lmbind(LIGHT1, 0) ;
lmbind(LIGHT2, 0) ;
lmbind(LMODEL, 0) ;
        for (i = 0 ; i < npolys ; i++) {
                c3f(redvec) ;
                /*c3f(blackvec) ;*/
                bgnclosedline() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                        v3f(P[i]->p[j]) ;
                }
                endclosedline() ;

}
}
  if (temp3 == 1)  {
lmbind(MATERIAL, 0) ;
lmbind(LIGHT0, 0) ;
lmbind(LIGHT1, 0) ;
lmbind(LIGHT2, 0) ;
lmbind(LMODEL, 0) ;

        for (i = 0 ; i < npolys ; i++) {
/*                      col_ind=(int)((V1[i]-mins[?])*scal);
                        col_ind=min(ncol-1,max(0,col_ind));
                c3f(&(color_palette[ col_ind][0])) ;
*/
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                col_ind=(int)((V2[i][j]-mins[2])*scal);
                col_ind=min(ncol-1,max(0,col_ind));
                c3f(&(color_palette[ col_ind][0])) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
        }
if(temp2 || temp0 )   {
lmbind(MATERIAL, 1) ;
if(light1->val)lmbind(LIGHT0, 1) ;else lmbind(LIGHT0,0);
if(light2->val)lmbind(LIGHT1, 2) ;else lmbind(LIGHT1,0);
if(light3->val)lmbind(LIGHT2, 3) ;else lmbind(LIGHT2,0);
lmbind(LMODEL, 1) ;
        for (i = 0 ; i < npolys ; i++) {
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                c3f(yelvec ) ;
                        if(temp2) n3f(N[i]) ;
                        if(temp0) n3f(N1[P[i]->s[j]]) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
        }

        swapbuffers() ;
make_reset();
}

def_simple_light_calc()
{
        lmdef(DEFMATERIAL, 1, 19, shinymaterial) ;
        lmdef(DEFLIGHT, 1, 10, sun1) ;
        lmdef(DEFLMODEL, 1, 0, NULL) ;
}


use_simple_light_calc()
{
        lmbind(MATERIAL, 1) ;
        lmbind(LIGHT0, 1) ;
        lmbind(LMODEL, 1) ;
}
```

```
int exitmode;

DO_JOB()
{
int mmm;
        mmm=80;
        c3f(backvec) ;
        clear() ;
        zclear() ;
while('exitmode) {
        pnl_dopanel() ;
        reshapeviewport();
        DrawObject();
        tumbl = tumble -> val;
        if(tumbl == 1)
        tum += mmm ;
        exitmode = quit->val;
}

}



  float kvadrat1[] = {-0.5,-0.5,-0.5};
  float kvadrat2[] = {-0.5,0.5,-0.5};
  float kvadrat3[] = {0.5,0.5,-0.5};
  float kvadrat4[] = {0.5,-0.5,-0.5};

  Drawbox()
  {
  char string1[5],string2[5],string3[5],string4[5];

  sprintf(string1,"%s","X");
  sprintf(string2,"%s","Y");
  sprintf(string3,"%s","Z");
  sprintf(string4,"%s","O");
   c3f(whitevec);


  cmov(0.50,-0.47,-0.50);
  charstr(string1);
  cmov(-0.50,0.53,-0.50);
  charstr(string2);
  cmov(-0.53,-0.53,0.53);
  charstr(string3);
  cmov(-0.47,-0.47,-0.47);
  charstr(string4);



   pushmatrix();
   bgnclosedline();
   v3f(kvadrat1);
   v3f(kvadrat2);
   v3f(kvadrat3);
   v3f(kvadrat4);
   endclosedline();
   popmatrix();
   pushmatrix();
   translate(0.0,0.0,1.0);
   bgnclosedline();
   v3f(kvadrat1);
   v3f(kvadrat2);
   v3f(kvadrat3);
   v3f(kvadrat4);
   endclosedline();
   popmatrix();

   pushmatrix();
   rotate(-900,'y');
   bgnclosedline();
   v3f(kvadrat1);
   v3f(kvadrat2);
   v3f(kvadrat3);
   v3f(kvadrat4);
   endclosedline();
   popmatrix();
   pushmatrix();
   translate(-1.0,0.0,0.0);
   rotate(-900,'y');
   bgnclosedline();
   v3f(kvadrat1);
   v3f(kvadrat2);
   v3f(kvadrat3);
   v3f(kvadrat4);
   endclosedline();
   popmatrix();
   }


init_palette()
{
int l,j;
float dxl;
for(l = 0;l<3;l++)
for(j = 0;j<ncol;j++)
color_palette[j][l] =0.0;

 for(l = 0;l<3;l++){
dxl= 4./ncol;
   if(l == 0) {
     for(j = ncol;j>= (int)(3*ncol/4)+1;j--){
      color_palette[j][l] = 1.0;
     }
     for(j = (int)(3*ncol/4);j>=ncol/2;j--){
      color_palette[j][l] = dxl*(j -(int)(ncol/2));
     }
   }
   if(l == 1) {
         for(j =(int) (ncol); j>= (int)(3*ncol/4);j--){
       color_palette[j][l] = 1 - dxl*(j -(int)( 3*ncol/4));
      }
       for(j =(int) (3*ncol/4); j>= (int)(ncol/4);j--){
        color_palette[j][l] = 1.0;
       }
      for(j =(int) (ncol/4); j>= 0;j--){
        color_palette[j][l] =  dxl*j;
    }

    }

   if(l == 2){
     for(j = ncol/2;j>= (int)(ncol/4);j--) {
       color_palette[j][l] =1.0 - dxl*(j-ncol/4);
      /* printf("%f %d %d \n", color_palette[j][l],l,j);*/
     }
```

```
    for(j = (int)(ncol/4);j>=0 ;j--) {
      color_palette[j][1] = 1.0;
    }

  }
 }
}

 get_palette()
{
int j;
FILE *fp, *fopen();
fp = fopen("color_palette","r");

  if(fp != NULL){
    fscanf(fp,"%d",&ncol);
      for(j = 0;j<ncol;j++)
       fscanf(fp,"%f %f %f",&color_palette[j][0],&color_palette[j][1],
              &color_palette[j][2]);


 fclose(fp);
scal=ncol/(maxs[2]-mins[2]);

    }

  else {ncol = maxcol; init_palette();}
 pnl_drawpanel();

.

jet_surface()


 ILE *fp, *fopen();


 p = fopen("lighting","r");
 f(fp !=NULL) {
 scanf(fp,"%f %f %f",&shinymaterial[1], &shinymaterial[2], &shinymaterial[3]);
 scanf(fp,"%f %f %f",&shinymaterial[5],&shinymaterial[6],&shinymaterial[7]);
 scanf(fp,"%f %f %f",&shinymaterial[9],&shinymaterial[10],&shinymaterial[11]);
 scanf(fp,"%f %f %f",&shinymaterial[13],&shinymaterial[14],&shinymaterial[15]);
 scanf(fp,"%f %f %f",&sun1[1],&sun1[2],&sun1[3]);
 mdef(DEFMATERIAL, 1, 19, shinymaterial) ;
 mbind(MATERIAL, 1) ;
 ndef(DEFLIGHT, 10,sun1);
 mbind(LIGHT0, 1);

 close(fp);
 nl_drawpanel();


 ave_position()

 loat siz,shx,shy,x_sc,y_sc,z_sc;
 loat dis,twi,the,ph;
 ILE *fp,*fopen();
        siz = size-> val;
        shx = shix ->val;
        shy = shiy ->val;
        x_sc = xsc->val;
        y_sc = ysc->val;
        z_sc = zsc->val;
        dis = distance ->val;
        twi = twist -> val;
        the = theta -> val;
        ph = phi ->val;

fp = fopen("pnl_position","w");
fprintf(fp,"%f %f %f %f %f %f\n",siz,shx,shy,x_sc,y_sc,z_sc);
fprintf(fp,"%f %f %f %f \n",dis,twi,the,ph);
fclose(fp);
pnl_drawpanel();


}


get_position()
{
int temp0,temp1,temp2,temp3;
float siz,shx,shy,x_sc,y_sc,z_sc;
float dis,twi,the,ph;
FILE *fp,*fopen();
fp = fopen("pnl_position","r");
if(fp != NULL) {fscanf(fp,"%f %f %f %f %f %f",&siz,&shx,&shy,&x_sc,&y_sc,&z_sc);
fscanf(fp,"%f %f %f %f",&dis,&twi,&the,&ph);
fclose(fp);

        size->val = siz;
        shix ->val = shx;
        shiy ->val = shy;
        xsc->val = x_sc;
        ysc->val = y_sc;
        zsc->val = z_sc;
        distance ->val = dis  ;
        twist -> val= twi  ;
        theta -> val=the  ;
        phi ->val= ph;
  }
pnl_drawpanel();

}



MakeVobj()
{
float siz,shx,shy,x_sc,y_sc,z_sc;
float x,y,z;
int theta1;
        siz = size -> val;
        shx = shix ->val;
        shy = shiy ->val;
        x_sc = xsc->val;
        y_sc = ysc->val;
        z_sc = zsc->val;


makeobj(Vobj1 = genobj() );
       loadmatrix(idmat) ;
        ortho(-2.0,2.0,-2.0,2.0,-2.0,2.0);
        /*perspective(fovy, 1.0, 0.1, k*k* Max+(1./ang)) ;*/
        closeobj();


makeobj(Vobj = genobj());
```

```
        loadmatrix(idmat) ;
        /*perspective(fovy, 1.0, 0.1, k*k* Max+(1./ang)) ;
        polarview(distance->val,0.0 ,0.0 ,0.0);*/
        ortho(-2.0,2.0,-2.0,2.0,-2.0,2.0);
        rotate((short)twist -> val,'y');
        rotate((short)theta -> val,'z');
        rotate((short)phi -> val,'x');
        scale(x_sc,y_sc,z_sc);
        rotate(tum,'y');
        translate(shx,shy,distance->val);
        scale(size->val,size->val,size->val);
closeobj();


makeobj(sphere = genobj());
for(theta1 = 0;theta1<3600;theta1 = theta1+300){
  rotate(300,'y');
  circ(0.0,0.0,1.0);
}
closeobj();

}

MakeLight(xmouse,ymouse)
Screencoord xmouse,ymouse;
{
long xor,yor;
float wx1,wy1,wz1;
getorigin(&xor,&yor);
xmouse = xmouse -xor;
ymouse = ymouse -yor;
mapw(Vobj,xmouse,ymouse,&wx1,&wy1,&wz1,&wx2,&wy2,&wz2);
sun1[5] = wx2;sun1[6] = wy2;sun1[7] = 5.0*Max;
lmdef(DEFLIGHT, 1, 10, sun1) ;
lmbind(LIGHT0, 1) ;
pushmatrix();
translate(wx2,wy2,wz2);
scale(0.01,0.01,0.01);
c3f(&(sun1[1]));
callobj(sphere);
popmatrix();

}

MakeLight1()
{
sun1[5] = xlight1->val;
sun1[6] = ylight1->val;
sun1[7] = zlight1->val;
lmdef(DEFLIGHT, 1, 10, sun1) ;
lmbind(LIGHT0, 1) ;
sun2[5] = xlight2->val;
sun2[6] = ylight2->val;
sun2[7] = zlight2->val;
lmdef(DEFLIGHT, 2, 10, sun2) ;
lmbind(LIGHT1, 2) ;
sun3[5] = xlight3->val;
sun3[6] = ylight3->val;
sun3[7] = zlight3->val;
lmdef(DEFLIGHT, 3, 10, sun3) ;
lmbind(LIGHT2, 3) ;

}
save_picture()
{
char savePict[50],*out_rgb,outStr[20];

long xor,yor,xsize,ysize;

getorigin(&xor,&yor);
getsize(&xsize,&ysize);
out_rgb = PNL_ACCESS(Typein,inputStr,str);
sprintf(savePict,"%s%s %d %d %d %d ","scrsave ",out_rgb,xor,
xor+xsize,yor,yor+ysize);

system(savePict);
}

#define RESET(v)   v->val = v->initval

make_reset()
{
RESET(distance);
RESET(theta);
RESET(phi);
RESET(twist);
RESET(size);
RESET(shix);
RESET(shiy);
RESET(xsc);
RESET(ysc);
RESET(zsc);
RESET(tumble);

pnl_drawpanel();
}



make_reset1()
{
loadmatrix(idmat);
getmatrix(OldMatrix);
}

ChangeStack()
{
loadmatrix(OldMatrix);
if(twist->active )rotate((short)twist -> val,'y');
if(theta->active)rotate((short)theta -> val,'z');
if(phi->active)rotate((short)phi -> val,'x');
if(xsc->active||ysc->active||zsc->active)scale(xsc->val,ysc->val,zsc->val);
if(size->active) scale(size->val,size->val,size->val);
getmatrix(OldMatrix);
}


main(argc, argv)
 int argc ;
 char * argv[] ;
{
        FILE *fp, *fopen();
        int i,j, temp2;
        short  val ;
        int dev;
        ncol = maxcol;

        negflag = 0;
```

```
        if (argc == 2)
                negflag = 1 ;
        printf("%d negflag \n",negflag);
        init_palette();
        ReadObject() ;

        keepaspect(1, 1) ;
        foreground() ;
        winopen("surface") ;
        RGBmode() ;
        doublebuffer() ;
        gconfig() ;

        lsetdepth(0, 0x7FFFFF) ;
        zbuffer(TRUE) ;

        mmode(MVIEWING) ;

        DefPanel() ;
        pnl_needredraw();
        def_simple_light_calc();
        loadmatrix(idmat);
        scale(size->val,size->val,size->val);
        /*translate(-1.0*shiftx,-1.0*shifty,0.0);*/
        getmatrix(OldMatrix);

        DrawObject();
        add_event(winget(),REDRAW,ANY,DO_JOB(),0);
        qdevice (REDRAW);
                }
```

```
#include <stdio.h>
#include <math.h>
#include <gl.h>
#include "panel.h"
#include <device.h>
#include<string.h>
#include "cpath.h"
#define  maxcol  256
#define MATRIXSIZE 16                /*size of matrix in the stack*/
#define LIGHTINDEX 10               /*size of sun file*/
#define   k     3.5
#define   ANY    -1
#define   SIZE   500000


float  color_palette[maxcol][3];
int ncol;
Object Vobj,sphere,Vobj1;
float wx2,wy2,wz2,scal_par;
typedef float   Vec[3] ;

typedef struct {
        int     p_nsides ;
        Vec     *p ;
        int      *s ;
} Polygon ;

 int    negflag ;
/*float         backvec[] = {0.8, 0.8, 0.8} ;*/
float   backvec[] = {0., 0., 0.} ;
float   blackvec[] = {0.0, 0.0, 0.0} ;
float   redvec[] = {1.0, 0.0, 0.0} ;
float   whitevec[] = {1.0, 1.0, 1.0} ;
float   yelvec[] = {1.0,1.0,0.0};


Matrix idmat = {1.0, 0.0, 0.0, 0.0,
                0.0, 1.0, 0.0, 0.0,
                0.0, 0.0, 1.0, 0.0,
                0.0, 0.0, 0.0, 1.0} ;
Matrix  OldMatrix ,NewMatrix;

Polygon *
MakePolygon(nsides)
 int nsides ;
{
        Polygon * p ;

        p = (Polygon *) malloc (sizeof(Polygon)) ;

        p -> p_nsides = nsides ;

        p -> p = (Vec *) calloc(nsides, sizeof(Vec )) ;
        p -> s = (int * ) malloc(3* sizeof(int )) ;

        return p ;
}

Vec     V[SIZE] ;
Polygon * P[SIZE] ;
Vec     N[SIZE],N1[SIZE] ;
int     npoints, npolys ;
Vec     mins = {1000000, 1000000, 1000000} ;
Vec     maxs = {-1000000, -1000000, -1000000} ;
```

```
float   V1[SIZE] ,V2[SIZE][3];

#define min(a, b) ((a) < (b) ? (a) : (b))
#define max(a, b) ((a) > (b) ? (a) : (b))

float Max ,scal;

#define         MKHSLIDER(v, n, b, t, i)\
        v = pnl_mkact(pnl_vslider) ;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v->w = 0.6;\
        v->h = 4.0;\
        v -> x = x ; v -> y = y ;\
        pnl_addact(v, p)


#define         LABEL(v,n)\
        v = pnl_mkact(pnl_label);\
        v->label = n;\
        v->x = x;\
        v->y = y;\
         pnl_addact(v,p)

#define         BUTTON(v, n, b, t, i)\
        v = pnl_mkact(pnl_radio_button) ;\
        v->labeltype = PNL_LABEL_BOTTOM;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = hl;v -> w = w;\
        pnl_addact(v, p)
#define         BUTTON2(v, n, b, t, i)\
        v = pnl_mkact(pnl_toggle_button) ;\
        v->labeltype = PNL_LABEL_BOTTOM;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = hl;v -> w = w;\
        pnl_addact(v, p)

#define         BUTTON3(v, n, b, t, i)\
        v = pnl_mkact(pnl_button) ;\
        v->labeltype = PNL_LABEL_BOTTOM;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = hl;v -> w = w;\
        pnl_addact(v, p)

Actuator * editPalette,* distance, * theta, * phi ,* model,*xlight1,*ylight1,
*zlight1,*xlight2,*ylight2,*zlight2,*xlight3,*ylight3,*zlight3,*surface,
*zsc, *map,*twist,*size, *shix,*shiy,*smooth,*xsc,*ysc,*reslight, *savePosition,
*getPosition, *myMouse, *savePict, *inputStr,*box, *reset,*quit, *l1,*l2,*l3,
*light1,*light2,*light3;
```

```
DefPanel()
{
        Panel   * p ;
        float   x = 0.0, y = 0.0, sep = 0.6,sep1=1.1 ,sep2 = 0.4;
        float   h1= 0.5, w = 1.0;

        p = pnl_mkpanel() ;
        p -> label = "Control Panel v.10" ;
        p -> ppu = 36.0 ;
        BUTTON3(quit,"QUIT",0.0,1.0,0.0);
        x = 4.5;y = quit->h+sep;
        BUTTON3(savePict,"SAVE PICTURE",0.0,1.0,0.0);
        y+= savePict->h+sep;
        inputStr = pnl_mkact(pnl_typein);
        inputStr->labeltype = PNL_LABEL_TOP_LEFT;
        inputStr ->label = "Save as: ";
        inputStr->x = 0.0;inputStr->y = y;
        pnl_addact(inputStr,p);
        y += inputStr->h+2*sep;
        x = 2.0;
        BUTTON2(box,"SHOW BOX",0.0,1.0,0.0);
        x = 5.0;
        BUTTON3(reset,"RESET",0.0,1.0,0.0);
        y +=reset->h+2*sep;
        x = 1.0;
        BUTTON3(savePosition,"Save Position",0.0,1.0,0.0);
        x = 4.5;
        BUTTON3(getPosition,"Get Position", 0.0,1.0,0.0);
        x = 8.0;
        BUTTON3(editPalette,"Edit Palette",0.0,1.0,0.0);
        x = 0.5;y+= w+sep;
        LABEL(12,"ROTATE");
        x = 3.0;
        LABEL(11,"TRANSLATE");
        x = 6.0;
        LABEL(13,"SCALE");
        y+=h1+sep;x = 0.0;
        MKHSLIDER(phi, "X ", -1800.0,1800.0, 0.0) ;
        x+=phi->w+0.3;
        MKHSLIDER(twist, "Y ", -1800.0,1800.0, 0.0) ;
        x+=phi->w+0.3;
        MKHSLIDER(theta, "Z ", -1800.0,1800.0, 0.0) ;
        x+=phi->w+0.6;
        MKHSLIDER(shix, " X", -1.0*Max, Max, 0.0) ;
        x+=theta->w+0.3;
        MKHSLIDER(shiy, " Y", -1.0*Max, Max, 0.0) ;
         x+=theta->w+0.3;
        MKHSLIDER(distance, " Z", -2.0,2.0,0.0) ;
        x+=theta->w+0.6;
        MKHSLIDER(xsc, "X ",-1.0, 1.0, 0.0) ;
        x+=theta->w+0.3;
        MKHSLIDER(ysc, "Y ",-1.0, 1.0, 0.0) ;
        x+=theta->w+0.3;
        MKHSLIDER(zsc, "Z ",-1.0, 1.0, 0.0) ;
        x+=theta->w+0.6;
        MKHSLIDER(size, "SIZE", -1.0, 1.0, 0.0) ;
        y+=size->h +1.5*sep;
        x = 0.5;
        BUTTON2(light1,"LIGHT",0.0,1.0,1.0);
        x = 3.5;
        BUTTON2(light2,"RED LIGHT",0.0,1.0,0.0);
        x = 6.5;
        BUTTON2(light3,"BLUE LIGHT",0.0,1.0,0.0);
        y+=h1+sep;
        x = 0.0;
        MKHSLIDER(xlight1,"X",-1.0,1.0,0.0);
        x+=theta->w+0.3;
        MKHSLIDER(ylight1,"Y",-1.0,1.0,0.0);
        x+=theta->w+0.3;
        MKHSLIDER(zlight1,"Z",-1.0,1.0,1.0);
        x+=theta->w+0.6;
        MKHSLIDER(xlight2,"X",-1.0,1.0,0.0);
        x+=theta->w+0.3;
        MKHSLIDER(ylight2,"Y",-1.0,1.0,0.0);
        x+=theta->w+0.3;
        MKHSLIDER(zlight2,"Z",-1.0,1.0,1.0);
        x+=theta->w+0.6;
        MKHSLIDER(xlight3,"X",-1.0,1.0,0.0);
        x+=theta->w+0.3;
        MKHSLIDER(ylight3,"Y",-1.0,1.0,0.0);
        x+=theta->w+0.3;
        MKHSLIDER(zlight3,"Z",-1.0,1.0,1.0);
        y+=zlight2->h+1.5*sep;
        x = 4.0;
        BUTTON3 (reslight,"RESET LIGHT",0.0,1.0,0.0);
        y+=h1+1.5*sep;x = 2.0;
        x = 2.0;
        BUTTON (smooth,"Gouraud Lighting",0.0,1.0,0.0);
        x = 6.0;
        BUTTON (map,"3D Map",0.0,1.0,0.0);
        y +=h1+1.5*sep;x = 2.0;
        BUTTON(surface, "Lighting", 0.0, 1.0, 0.0) ;
        x = 6.0;
        BUTTON(model, "Wireframe",  0.0, 1.0, 0.0) ;
        myMouse = pnl_mkact(pnl_mouse);
        pnl_addact(myMouse,p);


}
        float shiftx ,shifty,shiftz ;

ReadObject()
{

        int i, l, j, ku,nverts, indl[3], nsides ;
        Vec     tmp1, tmp2 ;

        scanf("%d %d\n", & npolys, & npoints) ;

/*
        fprintf(stderr, "%d points, %d polys\n", npoints, npolys) ;
        */
                for (j = 0 ; j < 3 ; j++) {
                        mins[j] =  le10;
                        maxs[j] =   -le10;
                }

        for (i = 0 ; i < npoints ; i++) {
                scanf("%f %f %f\n", &(V[i][0]), &(V[i][1]), &(V[i][2])) ;
        }


/*_____________________________________*/
        for (i = 0 ; i < npolys ; i++) {

                P[i] = MakePolygon(3) ;
                for (j = 0 ; j < 3 ; j++) {
```

```
                    scanf("%d", &ind1[j]) ;
                    l= ind1[j]-1;
                    P[i]->s[j] = l;
                    for (ku = 0 ; ku < 3 ; ku++) {
                    mins[ku] = min(mins[ku], V[l][ku]) ;
                    maxs[ku] = max(maxs[ku], V[l][ku]) ;
                     }
            }
        }

        Max = max(maxs[0]-mins[0], maxs[1]-mins[1]) ;
        Max = max(Max, maxs[2]-mins[2]) ;
        scal=ncol/(maxs[2]-mins[2]);
        shiftx = 0.5*(maxs[0]+mins[0]);
        shifty = 0.5*(maxs[1]+mins[1]);
        shiftz = 0.5*(maxs[2]+mins[2]);

        for (i = 0; i < npoints ; i++) {
        V[i][0] = V[i][0]- shiftx;
        V[i][1] = V[i][1] - shifty;
        V[i][2] = V[i][2] -  shiftz;
        }
        maxs[2] = maxs[2]-shiftz;
        mins[2] = mins[2]-shiftz;

        for (i = 0 ; i < npolys ; i++) {
                for (j = 0 ; j < 3 ; j++) {
                l=P[i]->s[j] ;
               for(ku = 0; ku< 3; ku++)
                (P[i]->p[j])[ku] = V[l][ku];
                V2[i][j] = V[l][2];
                }
           V1[i] = V2[i][0] + V2[i][1] +V2[i][2];
           V1[i] = V1[i]/3.;
        }


/*_____________________________________________*/
                /*mins[2]=0;*/

        scal_par = Max/2.0;


        for (i = 0 ; i < npolys ; i++) {
                VecSub(P[i]->p[0], P[i]->p[1], tmp1) ;
                VecSub(P[i]->p[2], P[i]->p[1], tmp2) ;
                VecCross(tmp1, tmp2, N[i]) ;
                VecNormalize(N[i]) ;
                if (negflag) {
                  if(N[i][0] < 0.)N[i][0] = - N[i][0];
                  if(N[i][1] < 0.)N[i][1] = - N[i][1];
                  if(N[i][2] < 0.)N[i][2] = - N[i][2];

                        }

             else{

                N[i][0] = - N[i][0] ;
                        N[i][1] = - N[i][1] ;
                        N[i][2] = - N[i][2] ;
                }
        }
        for (i = 0; i < npolys ; i++) {
        N1[i][0] = 0.0;
        N1[i][1] = 0.0;
        N1[i][2] = 0.0;
                        }

        for (i = 0; i < npolys ; i++) {
        for (j = 0; j< 3;j++) {
        N1[ P[i]->s[j] ][0]+= N[i][0];
        N1[ P[i]->s[j] ][1]+= N[i][1];
        N1[ P[i]->s[j] ][2]+= N[i][2];
        }
          }
        for (i = 0; i < npolys ; i++)
         VecNormalize(N1[i]);



}

VecNegate(a)
 Vec a ;
{
        a[0] = 0 - a[0] ;
        a[1] = 0 - a[1] ;
        a[2] = 0 - a[2] ;
}

VecSub(a, b, c)
 Vec a, b, c ;
{
        c[0] = a[0] - b[0] ;
        c[1] = a[1] - b[1] ;
        c[2] = a[2] - b[2] ;
}

VecCross(a, b, c)
 Vec a, b, c ;
{
        c[0] = a[1] * b[2] - b[1] * a[2] ;
        c[1] = - (a[0] * b[2] - b[0] * a[2]) ;
        c[2] = a[0] * b[1] - b[0] * a[1] ;
}

VecNormalize(a)
 Vec a ;
{
        float l ;
        l = (float) sqrt(a[0] * a[0] + a[1] * a[1] + a[2] * a[2]) ;

        a[0] /= l ;
        a[1] /= l ;
        a[2] /= l ;
}

float shinymaterial[] = { SPECULAR, 1.0, 0.2, 0.8,
                        EMISSION , 0.8,0.8,0.0,
                          DIFFUSE, 0.8, 0.0, 0.0,
                          AMBIENT, 0.2, 0.0, 0.0,
                          SHININESS, 3.0,
                          LMNULL } ;
float sun1[] = { LCOLOR,1.0, 1.0,1.0,
                POSITION, 0.0, 0.0, 1.0,0.0,
                LMNULL } ;
```

```
float sun2[] = { LCOLOR,1.0, 0.0,0.0,
                 POSITION, 0.0, 0.0, 1.0,0.0,
                 LMNULL } ;

float sun3[] = { LCOLOR,0.0, 0.0,1.0,
                 POSITION, 0.0, 0.0, 1.0,0.0,
                 LMNULL } ;

int tum ,tumbl;

DrawObject()
{
        Mouse * mymouse;
        short xmouse,ymouse;
        char  str[100];
        int i, j,col_ind ;
        get_surface();
        sprintf(str,"%s%s %d %s",PATH,"mycedit",ncol,"&");
        if(editPalette->val == 1.0) system(str);
        if(savePosition-> val == 1.0)save_position();
        if(getPosition->val == 1.0) get_position();
        if(savePict->val)save_picture();
        if(reset->val)make_reset();
        if(reslight->val)reset_light();
        reshapeviewport();
        c3f(backvec) ;
        clear() ;
        zclear() ;
        MakeVobj();
        callobj(Vobj1);
        MakeLight1();
        if(box->val){
          pushmatrix();
          scale(2.0,2.0,2.0);
          Drawbox();
          popmatrix();
        }

        ChangeStack();
        if (model->val == 1)  {
mbind(MATERIAL, 0) ;
mbind(LIGHT0, 0) ;
mbind(LIGHT1, 0) ;
mbind(LIGHT2, 0) ;
mbind(LMODEL, 0) ;
        for (i = 0 ; i < npolys ; i++) {
                c3f(redvec) ;
                bgnclosedline() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                        v3f(P[i]->p[j]);
                }
                endclosedline() ;



 if (map->val == 1)  {
mbind(MATERIAL, 0) ;
mbind(LIGHT0, 0) ;
mbind(LIGHT1, 0) ;
mbind(LIGHT2, 0) ;
mbind(LMODEL, 0) ;

        for (i = 0 ; i < npolys ; i++) {
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                col_ind=(int)((V2[i][j]-mins[2])*scal);
                col_ind=min(ncol-1,max(0,col_ind));
                c3f(&(color_palette[ col_ind][0])) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
        }
if(surface->val || smooth->val )   {
lmbind(MATERIAL, 1) ;
if(light1->val)lmbind(LIGHT0, 1) ;else lmbind(LIGHT0,0);
if(light2->val)lmbind(LIGHT1, 2) ;else lmbind(LIGHT1,0);
if(light3->val)lmbind(LIGHT2, 3) ;else lmbind(LIGHT2,0);
lmbind(LMODEL, 1) ;
        for (i = 0 ; i < npolys ; i++) {
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                c3f(yelvec ) ;
                        if(surface->val) n3f(N[i]) ;
                        if(smooth->val) n3f(N1[P[i]->s[j]]) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
        }


        swapbuffers() ;
make_initpanel();
}

def_simple_light_calc()
{
        lmdef(DEFMATERIAL, 1, 19, shinymaterial) ;
        lmdef(DEFLIGHT, 1, 10, sun1) ;
        lmdef(DEFLMODEL, 1, 0, NULL) ;
}


use_simple_light_calc()
{
        lmbind(MATERIAL, 1) ;
        lmbind(LIGHT0, 1) ;
        lmbind(LMODEL, 1) ;
}

int exitmode;

DO_JOB()
{
        c3f(backvec) ;
        clear() ;
        zclear() ;
while(!exitmode) {
        pnl_dopanel() ;
        reshapeviewport();
        DrawObject();
        exitmode = quit->val;
}

}
```

```
float kvadrat1[] = {-0.5,-0.5,-0.5};
float kvadrat2[] = {-0.5,0.5,-0.5};
float kvadrat3[] = {0.5,0.5,-0.5};
float kvadrat4[] = {0.5,-0.5,-0.5};



Drawbox()
{
char string1[5],string2[5],string3[5],string4[5];

sprintf(string1,"%s","X");
sprintf(string2,"%s","Y");
sprintf(string3,"%s","Z");
sprintf(string4,"%s","O");
 c3f(whitevec);


cmov(0.50,-0.47,-0.50);
charstr(string1);
cmov(-0.50,0.53,-0.50);
charstr(string2);
cmov(-0.53,-0.53,0.53);
charstr(string3);
cmov(-0.47,-0.47,-0.47);
charstr(string4);

 pushmatrix();
 bgnclosedline();
 v3f(kvadrat1);
 v3f(kvadrat2);
 v3f(kvadrat3);
 v3f(kvadrat4);
 endclosedline();
 popmatrix();
 pushmatrix();
 translate(0.0,0.0,1.0);
 bgnclosedline();
 v3f(kvadrat1);
 v3f(kvadrat2);
 v3f(kvadrat3);
 v3f(kvadrat4);
 endclosedline();
 popmatrix();

 pushmatrix();
 rotate(-900,'y');
 bgnclosedline();
 v3f(kvadrat1);
 v3f(kvadrat2);
 v3f(kvadrat3);
 v3f(kvadrat4);
 endclosedline();
 popmatrix();
 pushmatrix();
 translate(-1.0,0.0,0.0);
 rotate(-900,'y');
 bgnclosedline();
 v3f(kvadrat1);
 v3f(kvadrat2);
 v3f(kvadrat3);
 v3f(kvadrat4);
 endclosedline();
 popmatrix();
 }

init_palette()
{
int l,j;
float dxl;
for(l = 0;l<3;l++)
for(j = 0;j<ncol;j++)
color_palette[j][l] =0.0;

 for(l = 0;l<3;l++){
dxl= 4./ncol;
   if(l == 0) {
     for(j = ncol;j>= (int)(3*ncol/4)+1;j--){
      color_palette[j][l] = 1.0;
     }
     for(j = (int)(3*ncol/4);j>=ncol/2;j--){
      color_palette[j][l] = dxl*(j -(int)(ncol/2));
     }
   }
   if(l == 1) {
         for(j =(int) (ncol); j>= (int)(3*ncol/4);j--){
       color_palette[j][l] = 1 - dxl*(j -(int)( 3*ncol/4));
      }
       for(j =(int) (3*ncol/4); j>= (int)(ncol/4);j--){
        color_palette[j][l] = 1.0;
      }
      for(j =(int) (ncol/4); j>= 0;j--){
        color_palette[j][l] =  dxl*j;
    }

    }

   if(l == 2){
     for(j = ncol/2;j>= (int)(ncol/4);j--) {
       color_palette[j][l] =1.0 - dxl*(j-ncol/4);
      /* printf("%f %d %d \n", color_palette[j][l],l,j);*/
     }
     for(j = (int)(ncol/4);j>=0 ;j--) {
       color_palette[j][l] = 1.0;
     }

   }
 }
}

 get_palette()
{
int j;
FILE *fp, *fopen();
fp = fopen("color_palette","r");

  if(fp != NULL){
    fscanf(fp,"%d",&ncol);
      for(j = 0;j<ncol;j++)
       fscanf(fp,"%f %f %f",&color_palette[j][0],&color_palette[j][1],
              &ccolor_palette[j][2]);


 fclose(fp);
```

```
scal=ncol/(maxs[2]-mins[2]);

     }

   else {ncol = maxcol; init_palette();}
 pnl_drawpanel();

}

get_surface()
{

FILE *fp, *fopen();


fp = fopen("lighting","r");
if(fp !=NULL) {
fscanf(fp,"%f %f %f",&shinymaterial[1], &shinymaterial[2], &shinymaterial[3]);
fscanf(fp,"%f %f %f",&shinymaterial[5],&shinymaterial[6],&shinymaterial[7]);
fscanf(fp,"%f %f %f",&shinymaterial[9],&shinymaterial[10],&shinymaterial[11]);
fscanf(fp,"%f %f %f",&shinymaterial[13],&shinymaterial[14],&shinymaterial[15]);
fscanf(fp,"%f %f %f",&sun1[1],&sun1[2],&sun1[3]);
lmdef(DEFMATERIAL, 1, 19, shinymaterial) ;
lmbind(MATERIAL, 1) ;
lmdef(DEFLIGHT,1, 10,sun1);
lmbind(LIGHT0, 1);
}
fclose(fp);
pnl_drawpanel();
}

save_position()
{
FILE *fp,*fopen();
int i;
fp = fopen("pnl_position","w");
for(i = 0;i< MATRIXSIZE;i++)
fprintf(fp,"%f ",OldMatrix[i]);
for(i = 0;i< LIGHTINDEX;i++)
fprintf(fp,"%f ",sun1[i]);
for(i = 0;i< LIGHTINDEX;i++)
fprintf(fp,"%f ",sun2[i]);
for(i = 0;i< LIGHTINDEX;i++)
fprintf(fp,"%f ",sun3[i]);
fprintf(fp,"%f %f %f ",light1->val,light2->val,light3->val);
fclose(fp);


}


get_position()
{
int i;
FILE *fp,*fopen();
fp = fopen("pnl_position","r");
if(fp != NULL) {
   for(i = 0; i< MATRIXSIZE; i++)
     fscanf(fp,"%f ",&(OldMatrix[i]));
 for(i = 0;i< LIGHTINDEX;i++)
 fscanf(fp,"%f ",&(sun1[i]));
 for(i = 0;i< LIGHTINDEX;i++)
 fscanf(fp,"%f ",&(sun2[i]));
 for(i = 0;i< LIGHTINDEX;i++)
  fscanf(fp,"%f ",&(sun3[i]));
  fscanf(fp,"%f %f %f ",&(light1->val),&(light2->val),&(light3->val));
fclose(fp);
lmdef(DEFLIGHT,1, 10,sun1);
lmdef(DEFLIGHT,2, 10,sun2);
lmdef(DEFLIGHT,3, 10,sun3);
if(light1->val)lmbind(LIGHT0, 1) ;else lmbind(LIGHT0,0);
if(light2->val)lmbind(LIGHT1, 2) ;else lmbind(LIGHT1,0);
if(light3->val)lmbind(LIGHT2, 3) ;else lmbind(LIGHT2,0);
loadmatrix(OldMatrix);
pnl_drawpanel();

}

}



MakeVobj()
{
float siz,shx,shy,x_sc,y_sc,z_sc;
float x,y,z;
int theta1;
        siz = size -> val;
        shx = shix ->val;
        shy = shiy ->val;
        x_sc = xsc->val;
        y_sc = ysc->val;
        z_sc = zsc->val;


makeobj(Vobj1 = genobj() );
       loadmatrix(idmat) ;
        ortho(-2.0,2.0,-2.0,2.0,-2.0,2.0);
        closeobj();


makeobj(Vobj = genobj());
        callobj(Vobj1);
        rotate((short)twist -> val,'y');
        rotate((short)theta -> val,'z');
        rotate((short)phi -> val,'x');
        scale(x_sc,y_sc,z_sc);
        translate(shx,shy,distance->val);
closeobj();


makeobj(sphere = genobj());
for(theta1 = 0;theta1<3600;theta1 = theta1+300){
  rotate(300,'y');
  circ(0.0,0.0,1.0);
}
closeobj();

}

MakeLight(xmouse,ymouse)
Screencoord xmouse,ymouse;
{
long xor,yor;
float wx1,wy1,wz1;
getorigin(&xor,&yor);
xmouse = xmouse -xor;
ymouse = ymouse -yor;
```

```
mapw(Vobj,xmouse,ymouse,&wx1,&wy1,&wz1,&wx2,&wy2,&wz2);
sun1[5] = wx2;sun1[6] = wy2;sun1[7] = 5.0*Max;
lmdef(DEFLIGHT, 1, 10, sun1) ;
lmbind(LIGHT0, 1) ;
pushmatrix();
translate(wx2,wy2,wz2);
scale(0.01,0.01,0.01);
c3f(&(sun1[1]));
callobj(sphere);
popmatrix();

}

MakeLight1()
{
sun1[5] = xlight1->val;
sun1[6] = ylight1->val;
sun1[7] = zlight1->val;
lmdef(DEFLIGHT, 1, 10, sun1) ;
lmbind(LIGHT0, 1) ;
sun2[5] = xlight2->val;
sun2[6] = ylight2->val;
sun2[7] = zlight2->val;
lmdef(DEFLIGHT, 2, 10, sun2) ;
lmbind(LIGHT1, 2) ;
sun3[5] = xlight3->val;
sun3[6] = ylight3->val;
sun3[7] = zlight3->val;
lmdef(DEFLIGHT, 3, 10, sun3) ;
lmbind(LIGHT2, 3) ;

}
save_picture()
{
char savePict[50],*out_rgb,outStr[20];

long xor,yor,xsize,ysize;

getorigin(&xor,&yor);
getsize(&xsize,&ysize);
out_rgb = PNL_ACCESS(TypeIn,inputStr,str);
sprintf(savePict,"%s%s %d %d %d %d ","scrsave ",out_rgb,xor,
xor+xsize,yor,yor+ysize);

system(savePict);
}


make_reset()
{
callobj(Vobj1);
scale(pow(10,size->val),pow(10,size->val),pow(10,size->val));
getmatrix(OldMatrix);

}


#define RESET(v)   v->val = v->initval


reset_light()
{
RESET(xlight1);
RESET(ylight1);
RESET(zlight1);
RESET(xlight2);
RESET(ylight2);
RESET(zlight2);
RESET(xlight3);
RESET(ylight3);
RESET(zlight3);
RESET(light1);
RESET(light2);
RESET(light3);
pnl_drawpanel();
}

make_initpanel()
{
RESET(distance);
RESET(theta);
RESET(phi);
RESET(twist);
RESET(size);
RESET(shix);
RESET(shiy);
RESET(xsc);
RESET(ysc);
RESET(zsc);

pnl_drawpanel();
}


ChangeStack()
{
callobj(Vobj1);
if(twist->active )rotate((short)twist -> val,'y');
if(theta->active)rotate((short)theta -> val,'z');
if(phi->active)rotate((short)phi -> val,'x');
if(xsc->active||ysc->active||zsc->active)scale(pow(10,xsc->val),
pow(10,ysc->val),pow(10,zsc->val));
if(shix->active ||shiy->active ||distance ->active)
translate(shix->val,shiy->val,distance->val);
if(size->active ) scale(pow(10,size->val),pow(10,size->val),pow(10,size->val));
multmatrix(OldMatrix);
getmatrix(OldMatrix);
}



main(argc, argv)
 int argc ;
 char * argv[] ;
{
        FILE *fp, *fopen();
        int i,j;
        short  val ;
        int dev;
        ncol = maxcol;

        negflag == 0;
        if (argc == 2)
                negflag = 1 ;
        init_palette();
        ReadObject() ;

        keepaspect(1, 1) ;
```

```
        foreground() ;
        winopen("surface") ;
        RGBmode() ;
        doublebuffer() ;
        gconfig() ;

        lsetdepth(0, 0x7FFFFF) ;
        zbuffer(TRUE) ;

        mmode(MVIEWING) ;

        DefPanel() ;
        pnl_needredraw();
        def_simple_light_calc();
        callobj(Vobj1);
        scale(pow(10,size->val),pow(10,size->val),pow(10,size->val));
        getmatrix(OldMatrix);

        DrawObject();
        add_event(winget(),REDRAW,ANY,DO_JOB(),0);
        qdevice (REDRAW);
                }
```

```
#include<stdio.h>
#include<device.h>
#include<fmclient.h>
#include<gl.h>
#include <string.h>
#include<fmclient.h>
#define maxcol 10000
#define paramnum 5
#define PALETTE 2
#define LIGHTING 21
#define QUIT 5
#define BITS  31
#define PIECES 32
#define THRASH 33

 float blackvec[] = {0.0,0.0,0.0};
 float whitevec[] = {1.0,1.0,1.0};
float greenvec[] = {0.0,1.0,0.0};
float  bluevec[] = {0.0,0.0,1.0};
float  redvec[] = {1.0,0.0,0.0};
float grayvec[] = {0.7,0.7,0.7};
float rgbval[3];

short k,kOld ;/*number of the changing element of the palette*/
short m ; /*number of the changing element of light. param*/
long  xo,yo,xsize,ysize;
float color_palette[maxcol][3];
float lightvec[paramnum][3],shine;
long igid,exitmode, mode,moda;
int menu,menu1,menu2,menu3,submenu1,ncolmenu,ncolmenu1;
float range,rgbvalWidth,xRectSide,delta,dx,dx2,side;
 int ncol,index;

init_range(n)
int n;
{
mode = PALETTE;
ncol = n;
init_palette(ncol);
}


change_range(n)
int n;
{
int ncol_old;
float new_palette[maxcol][3];
int i,j,l;
int coef;

mode == PALETTE;
ncol_old = ncol;
ncol = n;
if(ncol_old < ncol){
  coef = ncol/ncol_old;
  for(j = 0;j<ncol_old;j++)
   for(l = 0;l< coef;l++)
    for(i = 0;i<3;i++){

        if(moda == PIECES)
         new_palette[j*coef+l][i] = color_palette[j][i];

        if(moda == PIECES)
         new_palette[j*coef+1][i] = color_palette[j][i];

        if(moda == BITS)
         new_palette[j*coef+l][i] = color_palette[j][i] + (l * (color_palette[j+1][i] -
                                    color_palette[j][i])/coef) ;

       }
      for (j = 0;j<ncol;j++)
       for(l = 0;l<coef;l++)
        for(i = 0;i<3;i++)
 color_palette[j][i] = new_palette[j][i];

  }

if(ncol_old>ncol){
 coef = ncol_old/ncol;
  for(j = 0;j<ncol;j++)
   for(i = 0;i<3;i++)
     new_palette[j][i] = color_palette[j*coef][i] ;

 for (j = 0;j<ncol;j++)
   for(i = 0;i<3;i++)
 color_palette[j][i] = new_palette[j][i];


}
  }

main(argc,argv)
int argc;
  char *argv[];
{
 FILE *fp1,*fp2,*fopen();

 ncol = atoi(argv[1]);
 initialize();
 init_palette(ncol);
 DrawObj();
  while(!exitmode){
    while(!qtest()){
      DrawObj();
      swapbuffers();
    }
    processinput();
  }
  winclose(igid);

}


initialize()
{
short i;
keepaspect(1,1);
foreground();
igid  = winopen("color_edit");
RGBmode();
doublebuffer();
gconfig();
ortho(0.0,1.0,0.0,1.0,-1.0,1.0);
c3f(whitevec);
clear();
qdevice(RIGHTMOUSE);
qdevice(LEFTMOUSE);
```

```
for(i= 0;i<3;i++)
rgbval[i] = 0.0;
k = -1;
mode = PALETTE;
moda = THRASH;
index =0;
MyMenus();
tie(RIGHTMOUSE,MOUSEX,MOUSEY);
tie(LEFTMOUSE,MOUSEX,MOUSEY);
getorigin(&xo,&yo);
getsize(&xsize,&ysize);
}



processinput()
{
short val;
short mx,my;
long menuval;
short i;

 while(qtest() ) {
   DrawObj();
 switch(qread(&val)) {
     case (REDRAW):
       reshapeviewport();
       getorigin(&xo,&yo);
       getsize(&xsize,&ysize);
         while(!qtest() ) {
           DrawObj();
           swapbuffers();
         }
       break;
      case (LEFTMOUSE):
       if(val){
         qread(&mx);
         qread(&my);
         make_rgbval(mx,my);
         DrawObj();
         swapbuffers();

        }
        break;
      case(RIGHTMOUSE):
         menuval  = dopup(menu1);
          switch(menuval){
          case 1:
             if (mode == PALETTE) init_palette(ncol);
             else init_lighting();
             DrawObj();
             swapbuffers();
             break;
          case 2:
             if (mode == PALETTE) get_palette();
             else get_lighting();
             DrawObj();
             swapbuffers();
             break;
          case 3:
            if (mode == PALETTE) save_palette();
             else save_lighting();
             DrawObj();
             swapbuffers();
             break;
      case LIGHTING:
           mode = LIGHTING;
           init_lighting();
           DrawObj();
           swapbuffers();
           break;

       case BITS:
           mode = PALETTE;
               moda = BITS;
           DrawObj();
           swapbuffers();
           break;
       case PIECES:
           mode = PALETTE;
               moda = PIECES;
           DrawObj();
           swapbuffers();
           break;
       case THRASH:
           mode = PALETTE;
               moda = THRASH;
           DrawObj();
           swapbuffers();
           break;

        case 41:
                  if(mode == PALETTE)
                  save_palette();
                  else save_lighting();
                  exitmode = 1;
                  break;
         case 42:
                   exitmode = 1;
                   break;
         case 43:
                   exitmode = 0;
                   break;
        default:
          break;
       }

     }

   }
}



init_palette(ncol)
int ncol;
{
int i,j;
float dx1;
for(i = 0;i<3;i++)
for(j = 0;j<ncol;j++)
color_palette[j][i] =0.0;
```

```
kOld = -1;
index = 0;
 for(i = 0;i<3;i++){
dx1= 4./ncol;
   if(i == 0) {
     for(j = ncol;j>= (int)(3*ncol/4)+1;j--){
      color_palette[j][i] = 1.0;
     }
     for(j = (int)(3*ncol/4);j>=ncol/2;j--){
      color_palette[j][i] = dx1*(j -(int)(ncol/2));
     }

   }
   if(i == 1) {
         for(j =(int) (ncol); j>= (int)(3*ncol/4);j--){
       color_palette[j][i] = 1 - dx1*(j -(int)( 3*ncol/4));
      }
       for(j =(int) (3*ncol/4); j>= (int)(ncol/4);j--){
        color_palette[j][i] = 1.0;
       }

      for(j =(int) (ncol/4); j>= 0;j--){
        color_palette[j][i] =  dx1*j;
       }

    }

   if(i == 2){
     for(j = ncol/2;j>= (int)(ncol/4);j--) {
       color_palette[j][i] =1.0 - dx1*(j-ncol/4);
     }
     for(j = (int)(ncol/4);j>=0 ;j--) {
       color_palette[j][i] = 1.0;
     }

   }
 }

}



DrawObj()
{
 fmfonthandle f;
 fmfonthandle fsized;
float size;
short i,k1;
float x[maxcol],y[maxcol],x1[6],y1[6];
float yRectSide,rgbvalSide,rgbvalHeight,col[3];
char str1n[30], *lightstr[paramnum];
delta = 1./20.;
 c3f(whitevec);
clear();

if(mode == PALETTE ){
/*palette*/

dx = (1.0- 2*delta)/(ncol);
  for (i = 0;i<ncol;i++){
   x[i] =delta+dx*i;
   y[i] =delta;
   c3f(&(color_palette[i][0]));
   rectf(x[i],y[i],x[i]+dx,y[i]+delta);
    if(i == k){
     c3f(blackvec);
     rect(x[i],y[i],x[i]+dx,y[i]+delta);
    }

  }
 }
else if(mode == LIGHTING){

size = (float)xsize/40.0;
fminit();
f = fmfindfont("Times-Roman");
fsized = fmscalefont(f,size);
fmsetfont(fsized);

  lightstr[0] = "specular";
  lightstr[1] = "emission";
  lightstr[2] = "diffuse";
  lightstr[3] = "ambient";
  lightstr[4] = "light";
      dx2 = (1.0 - 2*delta)/(5.0*paramnum);
      side = 4*dx2;
      for(i = 0;i<paramnum;i++){
        c3f(&(lightvec[i][0]));
        rectf(delta+(side+dx2)*i,delta,delta+side*(i+1)+dx2*i,2*delta);
        c3f(blackvec);
        rect(delta+(side+dx2)*i,delta,delta+side*(i+1)+dx2*i,2*delta);
        cmov2(delta + (side+dx2)*i, delta/2.0);
        fmprstr(lightstr[i]);
       }
      }
/* draw RGB rects*/

  xRectSide = 1./5. - delta;
  yRectSide = 1. - 4*delta;

  for(i = 1; i<4; i++){

  c3f(grayvec);
  rectf(delta*i+xRectSide*(i -1),3*delta,delta*i+xRectSide*i,3*delta+yRectSide);
  c3f(whitevec);
  rectf(delta*(i+1)+xRectSide*(i - 1),4*delta,delta*(i-1)+
  xRectSide*i,2*delta+yRectSide);
 c3f(blackvec);



    range = yRectSide - 2*delta;
  rgbvalSide = range/100.;
  rgbvalWidth = xRectSide - 2*delta;
  rgbvalHeight = range*rgbval[i-1]+4*delta;
  for(k1 = 0;k1< 3;k1++)
    col[k1] = 0.0;
   col[i-1] = 1.0;
   c3f(col);
   rectf(delta*(i+1)+xRectSide*(i - 1),rgbvalHeight,delta*(i+1)+xRectSide*(i - 1) +
    rgbvalWidth,rgbvalHeight-rgbvalSide);
 rect(delta*i+xRectSide*(i - 1),3*delta,delta*i+xRectSide*i,3*delta+yRectSide);
 rect(delta*(i+1)+xRectSide*(i - 1),4*delta,delta*(i-1)+xRectSide*i,2*delta+yRectSide);



 }
```

```
*sample rectangle*/
 c3f(rgbval);
rectf(1 - delta - 2*xRectSide,3*delta,1 - delta,3*delta+yRectSide);
 c3f(blackvec);
 rect(1 - delta - 2*xRectSide,3*delta,1 - delta,3*delta+yRectSide);



}



ake_rgbval(x,y)
hort x,y;

 float z1,z2;
 int b1,b2;
 int ncol_old , jump;
 float x1,y1,y2;  short i,OK,j;
 OK = 0;

 x1= (float)(x-xo)/(float)xsize;
 y1= (float)(y-yo)/(float)ysize;


f(y1 > 2*delta) {

   y1 = (y1 - (4*delta))/range;
   i = (int)(x1/(delta + xRectSide));
   if((x1 - (delta*(i+2) + xRectSide*i)) >0 && (x1 - (delta*(i+2) + xRectSide*i))
 < rgbvalWidth) OK = 1;
   if(y1 > 0 && y1 < 1.0 && OK == 1) rgbval[i] = y1;
}
 /*choosing k - th el of the palette*/
lse {if(mode == PALETTE) {
     if(y1 > delta && y1 < 2*delta && x1 > delta && x1< 1.0 - delta){

            kOld = k;
            k = (int)((x1 - delta)/dx);
             index++;
            printf("%d k  %d kOld \n",k,kOld);
            if(moda == PIECES && index%2 == 0 ){

              if(kOld < k && k>=0 && kOld>=0) for(j = kOld;j<k+1;j++)
                             for(i = 0;i<3;i++)
                               color_palette[j][i] = rgbval[i];

              else if(k>=0 && kOld >=0)for(j = k;j<kOld+1;j++)
                             for(i = 0;i<3;i++)
                               color_palette[j][i] = rgbval[i];


            }

          if(moda == THRASH && index%2 == 0)  {

          ncol_old = ncol;
            if(k>kOld){
            z2 = (float)ncol_old/(float)k;
            z1 = (float)ncol_old/(float)kOld;

             jump = k-kOld;
            }
            if(k < kOld) {
            z1 = (float)ncol_old/(float)k;
            z2 = (float)ncol_old/(float)kOld;

             jump = kOld-k;
            }


            ncol = (ncol_old *ncol_old)/jump;
            b1 = (int)((float)ncol/z1);
            b2 = (int)((float)ncol/z2);
            for (j = ncol;j> b2; j--)
              for(i = 0;i<3;i++)
                color_palette[j][i] = redvec[i];

                 m = ncol_old;
             for(j = b2;j>b1; j--){
                  m--;
                 for (i = 0;i<3;i++)
               color_palette[j][i] = color_palette[m][i];

            }


             for(j = 0;j<=b1 ;j++)
              for (i = 0;i<3;i++)
               color_palette[j][i] = bluevec[i];


        }
              if(moda == BITS)for(i = 0;i<3;i++){
                                color_palette[k][i] = rgbval[i];
              }
            }
       }

     if(mode == LIGHTING) {
/* choosing the m-th element of light. param*/
     m  = (int)((x1 - delta)/(dx2+side));
     if((delta+side*(m+1)+dx2*m - x1)<=side && x1<=delta+side*(m+1)+dx2*m && y1> delta)
for(i = 0;i<3;i++)lightvec[m][i] = rgbval[i];
     }
   }

}

get_palette()
{
int j;
FILE *fp, *fopen();
fp = fopen("color_palette","r");
  if(fp != NULL){
    fscanf(fp,"%d",&ncol);
     for(j = 0;j<ncol;j++)
       fscanf(fp,"%f %f %f",&color_palette[j][0],&color_palette[j][1],
              &color_palette[j][2]);

        k = -1;
 fclose(fp);
     }

   else init_palette(ncol);
   k = -1;
}

save_palette()
```

```
{
int j;
FILE *fp, *fopen();

fp = fopen("color_palette","w");
fprintf(fp,"%d\n",ncol);
for(j = 0;j<ncol;j++)
fprintf(fp,"%f %f %f\n",color_palette[j][0],color_palette[j][1],
                color_palette[j][2]);
fclose(fp);


}

get_lighting()
{
int j;
FILE *fp, *fopen(),
fp = fopen("lighting","r");
  if(fp != NULL){
     for(j = 0;j<paramnum;j++)
       fscanf(fp,"%f %f %f",&lightvec[j][0],&lightvec[j][1],
               &lightvec[j][2]);

        k = -1;
 fclose(fp);
     }

   else init_lighting();
   k = -1;
}


save_lighting()
{
int j;
FILE *fp, *fopen();

fp = fopen("lighting","w");

for(j = 0;j<paramnum;j++)
fprintf(fp,"%f %f %f\n",lightvec[j][0],lightvec[j][1],
               lightvec[j][2]);
fclose(fp);


}

init_lighting()
{
                /*SPECULAR*/              /*DIFFUSE*/
              lightvec[0][0] = 1.0;     lightvec[2][0] = 1.0;
              lightvec[0][1] = 0.0;     lightvec[2][1] = 0.6;
              lightvec[0][2] = 0.0;     lightvec[2][2] = 0.0;
                 /*EMISSION*/              /*AMBIENT*/
              lightvec[1][0] = 1.0;      lightvec[3][0] = 1.0;
              lightvec[1][2] = 0.2;      lightvec[3][1] = 0.0;
             lightvec[1][3] = 0.0;      lightvec[3][2] = 0.0;

               /*LIGHT*/
              lightvec[4][0] = 1.0;
              lightvec[4][1] = 1.0;
              lightvec[4][2]  = 1.0;
}
```

```
MyMenus()
{
menu2 = defpup("SAVE? %t| YES %x41 | NO %x42|CANCEL %x43 ");
ncolmenu = defpup("COLOR RANGE %t %F|8 %x8|16 %x16|32 %x32|64 %x64|256 %x256",init_range)
;
ncolmenu1 = defpup("COLOR RANGE %t %F|8 %x8|16 %x16|32 %x32|64 %x64|256 %x256",change_ran
ge);

submenu1 = defpup("PALETTE %t|BITS %x31| PIECES %x32| THRASH %x33");
menu = defpup("CEDIT %t|LIGHTING %x21|PALETTE %m",submenu1);
addtopup(menu,"COLOR RANGE %m",ncolmenu1);
menu1=defpup("CEDIT %t|INIT %m", ncolmenu);
addtopup(menu1,"GET|SAVE|CHANGE MODE %m",menu);
addtopup(menu1,"QIUT %m",menu2);

}
```

```
#include<stdio.h>
#include<device.h>
#include<fmclient.h>
#include<gl.h>
#include <string.h>
#include "cpatn.h"
#include "panel.h"


#define N 1/*number of buttons on the control panel*/
#define         BUTTON(v, n, b, t, i)\
        v = pnl_mkact(pnl_button) ;\
        v -> label = n ;\
        v->labeltype = PNL_LABEL_BOTTOM;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> w = 1.0;\
        v -> h = 0.5;\
        v -> x = x ; v -> y = y


Actuator *inputStr;
anel *MyDialBox(labelStr,Button,butAct)
har *labelStr,*Button[N];
ctuator *butAct[N];

nt i,n;
ctuator *mylabel;
loat sep,x,y;
anel *panelPtr;
ep = 0.5;
anelPtr = pnl_mkpanel();
anelPtr->x = 600;
anelPtr->y = 500;
anelPtr->visible = TRUE;  /* panel is initialised invisible'*/
                         /* to make it visible say panelPtr->visible = TRUE */
                          /*  and pnl(fixpanel(panelPtr);*/

 = 0;
or(i = 0;i<N;i++)
f(Button[i]!=0){
utton[n] = Button[i];
++;

 += 2*sep;
 = 0.0;
nputStr = pnl_mkact(pnl_typein);
nl_addact(inputStr,panelPtr);
 = 12*sep;
 = inputStr->y - 2*sep;
or(i = 0;i<n;i++){
UTTON(butAct[i],Button[i],0.0,1.0,0.0);
-=sep+butAct[i]->w;
nl_addact(butAct[i],panelPtr);

/label = pnl_mkact(pnl_label);
/label->label = labelStr;
/label->y = inputStr->y+2*sep;
/label->x = sep;
nl_addact(mylabel,panelPtr);
eturn panelPtr;


ain()
{
FILE *fp, *fopen();
char geodata[100], pluc.[100];
char *mystring;
int i;
long igid1;
short mx,my;
char *But[N];
Actuator *but[N];
Panel *dial;
int mode = 1;
float x =0.0;float y = 0.0;
foreground();
noport();
keepaspect(1,1);
igid1 = winopen("");
doublebuffer();
RGBmode();
gconfig();
But[0] = "CONTINUE";
But[1] = "SELECT";
dial = MyDialBox("INPUT FILE NAME",But,but);
pnl_needredraw();
 fp = fopen("geoname","w");       /*erase geoname*/
  fclose(fp);
 while('(but[0]->val)){
   pnl_userredraw();
   pnl_dopanel();
    mystring = PNL_ACCESS(Typein,inputStr,str);
  }
pnl_delpanel(dial);
  fp = fopen("geoname","w");
 fprintf(fp,"%s",mystring);
 fclose(fp);

/*winclose(igid1);*/
}
```

```
#include <stdio.h>
#include <math.h>
#include <gl.h>
#include "panel.h"
#include <device.h>
#define  ncol  12
#define   k     3.5
#define   ANY    -1
#define   SIZE   100000


float  legend[ncol][3] = {
{0.0 ,0.0,0.8},
{0.0 ,0.0,1.0},
{0.0 ,0.6,0.6},
{0.0 ,0.8,0.0},
{0.0 ,1.0,0.0},
{0.2 ,0.9,0.0},
{1.0 ,1.0,0.0},
{1.0 ,0.8,0.0},
{1.0 ,0.6,0.0},
{1.0 ,0.4,0.0},
{1.0 ,0.2,0.0},
{1.0 ,0.0,0.0},
};

typedef float   Vec[3] ;

typedef struct {
        int     p_nsides ;
        Vec     ** p ;
        int      *s ;
} Polygon ;

int     negflag ;
/*float         backvec[] = {0.8, 0.8, 0.8} ;*/
float   backvec[] = {0., 0., 0.} ;
float   blackvec[] = {0.0, 0.0, 0.0} ;
float   redvec[] = {1.0, 0.0, 0.0} ;
float   whitevec[] = {1.0, 1.0, 1.0} ;

float idmat[] = {1.0, 0.0, 0.0, 0.0,
                 0.0, 1.0, 0.0, 0.0,
                 0.0, 0.0, 1.0, 0.0,
                 0.0, 0.0, 0.0, 1.0} ;

Polygon *
MakePolygon(nsides)
 int nsides ;
{
        Polygon * p ;

        p = (Polygon *) malloc (sizeof(Polygon)) ;

        p -> p_nsides = nsides ;

        p -> p = (Vec **) calloc(nsides, sizeof(Vec *)) ;
        p -> s = (int * ) malloc(3* sizeof(int )) ;

        return p ;
}

Vec     V[SIZE] ;
Polygon * P[SIZE] ;

Vec     N[SIZE],N1[SIZE] ;
int     npoints, npolys ;
Vec     mins = {1000000, 1000000, 1000000} ;
Vec     maxs = {-1000000, -1000000, -1000000} ;
float   mi = 1000000;
float   ma = -1000000;
float   V1[SIZE] ,V2[SIZE][3],V3[SIZE][3];

#define min(a, b)  ((a) < (b) ? (a) : (b))
#define max(a, b)  ((a) > (b) ? (a) : (b))

float Max ,scal,fovy;

#define         MKHSLIDER(v, n, b, t, i)\
        v = pnl_mkact(pnl_hslider) ;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        pnl_addact(v, p) ;\
        y += v -> h + sep

#define         MKHSLIDER1(v, n, b, t, i)\
        v = pnl_mkact(pnl_hslider) ;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = h1;v -> w = w;\
        pnl_addact(v, p) ;\
        x += v -> w + sep1

#define         BUTTON(v, n, b, t, i)\
        v = pnl_mkact(pnl_radio_button) ;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = h1;v -> w = w;\
        pnl_addact(v, p) ;\
        y += v -> h + sep;\
        x += v -> w - 4*sep-w
#define         BUTTON1(v, n, b, t, i)\
        v = pnl_mkact(pnl_radio_button) ;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = h1;v -> w = w;\
        pnl_addact(v, p) ;\
        x += v -> h + 3*sep
#define         BUTTON2(v, n, b, t, i)\
        v = pnl_mkact(pnl_toggle_button) ;\
        v -> label = n ;\
        v -> minval = b ;\
        v -> maxval = t ;\
        v -> val = i ;\
        v -> x = x ; v -> y = y ;\
        v -> h = h1;v -> w = w;\
        pnl_addact(v, p) ;\
```

```
        y += v -> h + sep;\
        x += v -> w - 4*sep-w

Actuator * distance, * theta, * phi ,* model,* surface, *map,
*twist,*angle,*shix,*shiy,*smooth,*xsc,*ysc,*tumble;

DefPanel()
{
        Panel   * p ;
        float   x = 0.0, y = 0.0, sep = 0.7,sep1=1.1 ;
        float   h1= 0.5, w = 1.0;

        p = pnl_mkpanel() ;
        p -> label = "Control Panel v.10" ;
        p -> ppu = 36.0 ;
MKHSLIDER(distance, "DISTANCE", k*Max, k*k*Max, k* Max) ;
        MKHSLIDER(theta, "Z ROTATION", 0.0, 3600.0, 0.0) ;
        MKHSLIDER(phi, "X ROTATION", 0.0, 3600.0, 0.0) ;
        MKHSLIDER(twist, "Y ROTATION", 0.0, 3600.0, 0.0) ;
        MKHSLIDER(angle, "SIZE", 0.5,0.01, 0.3) ;
        MKHSLIDER(shix, "SHIFT_X", -1.0*Max, Max, 0.0) ;
        MKHSLIDER(shiy, "SHIFT_y", -1.0*Max, Max, 0.0) ;
        MKHSLIDER(xsc, "XSCALE",.1,10, 1.0) ;
        MKHSLIDER(ysc, 'YSCALE",.1,10 ,1.0) ;
        BUTTON1(model, "Model",  0.0, 1.0, 0.0) ;
        BUTTON(surface, "Surface", 0.0, 1.0, 0.0) ;
        BUTTON1 (map,"Map",0.0,1.0,0.0);
        BUTTON (smooth,"Smooth",0.0,1.0,0.0);
        BUTTON2 (tumble,"Tumble",0.0,1.0,0.0);


}
        float sh.ftx ,shifty,tempo ;

ReadObject()
{

        int i, j, ku,nverts, index[3], nsides ;
        Vec     tmp1, tmp2 ;

        scanf("%d %d\n", & npolys, & npoints) ;

/*
        fprintf(stderr, "%d points, %d polys\n", npoints, npolys) ;
        */
                for (j = 0 ; j < 3 ; j++) {
                        mins[j] =  1e10;
                        maxs[j] =   -1e10;
                }

        for (i = 0 ; i < npoints ; i++) {
                scanf("%f %f %f\n", &(V[i][0]), &(V[i][1]), &(V[i][2])) ;
        }

/*Lala: I changed here */
        for (i = 0; i < npoints ; i++) {
        tempo=V[i][1];
        V[i][0] = V[i][0]- shiftx;
        V[i][1] = tempo - shifty;
        }



        for (i = 0 ; i < npolys ; i++) {
                P[i] = MakePolygon(3) ;
                for (j = 0 ; j < 3 ; j++) {
                        scanf("%d", &index[j]) ;
                        P[i]->s[j] = index[j] -1 ;
                        P[i]->p[j] = V + (index[j] - 1) ;
                        V2[i][j] = V[index[j] -1][2];
                for (ku = 0 ; ku < 3 ; ku++) {
                        mins[ku] = min(mins[ku], V[index[j]-1][ku]) ;
                        maxs[ku] = max(maxs[ku], V[index[j]-1][ku]) ;
                V3[i][j] += (V[index[j]-1][ku] * V[index[j]-1][ku]);
                }
                        V3[i][j] = sqrt(V3[i][j]);
                        mi = min(mi,V3[i][j]);
                        ma = max(ma,V3[i][j]);
                }
           V1[i] = V2[i][0] + V2[i][1] +V2[i][2];
           V1[i] = V1[i]/3.;
        }
/*


fprintf(stderr, "bbx: %f - %f\n", mins[0], maxs[0]) ;
fprintf(stderr, "bby: %f - %f\n", mins[1], maxs[1]) ;
fprintf(stderr, "bbz: %f - %f\n", mins[2], maxs[2]) ;
*/
        /*mins[2]=0;*/
if(ma >= mi )
scal=ncol/(ma-mi);
else scal = ncol/ma;

Max = max(maxs[0]-mins[0], maxs[1]-mins[1]) ;
Max = max(Max, maxs[2]-mins[2]) ;
shiftx = 0.5*(maxs[0]+mins[0]);
shifty = 0.5*(maxs[1]+mins[1]);

for (i = 0 ; i < npolys ; i++) {
        VecSub(P[i]->p[0], P[i]->p[1], tmp1) ;
        VecSub(P[i]->p[2], P[i]->p[1], tmp2) ;
        VecCross(tmp1, tmp2, N[i]) ;
        VecNormalize(N[i]) ;
        if (negflag) {} else{
        N[i][0] = - N[i][0] ;
                N[i][1] = - N[i][1] ;
                N[i][2] = - N[i][2] ;
        }
}
for (i = 0; i < npolys ; i++) {
N1[i][0] = 0.0;
N1[i][1] = 0.0;
N1[i][2] = 0.0;
           }

for (i = 0; i < npolys ; i++) {
for (j = 0; j< 3;j++) {
N1[ P[i]->s[j] ][0]+= N[i][0];
N1[ P[i]->s[j] ][1]+= N[i][1];
N1[ P[i]->s[j] ][2]+= N[i][2];
}
  }
for (i = 0; i < npolys ; i++)
 VecNormalize(N1[i]);
```

```
}

VecNegate(a)
 Vec a ;
{
        a[0] = 0 - a[0] ;
        a[1] = 0 - a[1] ;
        a[2] = 0 - a[2] ;
}

VecSub(a, b, c)
 Vec a, b, c ;
{
        c[0] = a[0] - b[0] ;
        c[1] = a[1] - b[1] ;
        c[2] = a[2] - b[2] ;
}

VecCross(a, b, c)
 Vec a, b, c ;
{
        c[0] = a[1] * b[2] - b[1] * a[2] ;
        c[1] = - (a[0] * b[2] - b[0] * a[2]) ;
        c[2] = a[0] * b[1] - b[0] * a[1] ;
}

VecNormalize(a)
 Vec a ;
{
        float l ;
        l = (float) sqrt(a[0] * a[0] + a[1] * a[1] + a[2] * a[2]) ;

        a[0] /= l ;
        a[1] /= l ;
        a[2] /= l ;
}

int tum ,tumbl;
DrawObject()
{
        int i, j,col_ind ;
        int temp0,temp1,temp2,temp3,tumbl;
        float ang,shx,shy,x_sc,y_sc;
        temp1=model -> val;
        temp2=surface -> val;
        temp0=smooth -> val;
        temp3=map -> val;
        ang = angle -> val;
        shx = shix ->val;
        shy = shiy ->val;
        x_sc = xsc->val;
        y_sc = ysc->val;

        reshapeviewport();
        c3f(backvec) ;
        clear() ;
        zclear() ;
        fovy=(atan(ang)*1800)/PI;
        fovy=2*fovy;


        loadmatrix(idmat) ;
        perspective(fovy, 1.0, 0.1, k*k* Max+(1./ang)) ;
        polarview(distance->val,0.0 ,0.0 ,0.0);
        rotate((short)twist -> val,'y');
        rotate((short)theta -> val,'z');
        rotate((short)phi -> val,'x');
        scale(x_sc,y_sc,1.0);
        rotate(tum,'y');

/*      rotate (900,'z');*/


        translate(shx,shy,0.0);
        Drawbox();
         translate(-1.0*shiftx,-1.0*shifty,0.0);
        if (temp1 == 1)  {
lmbind(MATERIAL, 0) ;
lmbind(LIGHT0, 0) ;
lmbind(LMODEL, 0) ;
        for (i = 0 ; i < npolys ; i++) {
                c3f(redvec) ;
                /*c3f(blackvec) ;*/
                bgnclosedline() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                        v3f(P[i]->p[j]) ;
                }
                endclosedline() ;

}
}
  if (temp3 == 1)  {
lmbind(MATERIAL, 0) ;
lmbind(LIGHT0, 0) ;
lmbind(LMODEL, 0) ;

        for (i = 0 ; i < npolys ; i++) {
/*                      col_ind=(int)((V1[i]-mins[2])*scal);
                        col_ind=min(ncol-1,max(0,col_ind));
                c3f(&(legend[ col_ind][0])) ;
*/
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                col_ind=(int)((V3[i][j]-mi)*scal);
                col_ind=min(ncol-1,max(0,col_ind));
                c3f(&(legend[ col_ind][0])) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
        }
if(temp2 || temp0 )   {
lmbind(MATERIAL, 1) ;
lmbind(LIGHT0, 1) ;
lmbind(LMODEL, 1) ;
        for (i = 0 ; i < npolys ; i++) {
                bgnpolygon() ;
                for (j = 0 ; j < P[i] -> p_nsides ; j++) {
                c3f(&(legend[ 7][0])) ;
                        if(temp2) n3f(N[i]) ;
                        if(temp0) n3f(N1[P[i]->s[j]]) ;
                        v3f(P[i]->p[j]) ;
                }
                endpolygon() ;
        }
```

```
        }
        swapbuffers() ;
}

float shinymaterial[] = { SPECULAR, 1.0, 0.0, 0.0,
                          EMISSION , 1.0,.20,0.0,
                            DIFFUSE, 1.0, 0.6, 0.0,
                            AMBIENT, 1.0, 0.0, 0.0,
                            SHININESS, 3.0,
                            LMNULL ) ;
float sun[] = { POSITION, 0.0, 0.0, 1.0,0.0,
                            LMNULL ) ;
def_simple_light_calc()
{
        lmdef(DEFMATERIAL, 1, 19, shinymaterial) ;
        lmdef(DEFLIGHT, 1, 6, sun) ;
        lmdef(DEFLMODEL, 1, 0, NULL) ;
}

use_simple_light_calc()
{
        lmbind(MATERIAL, 1) ;
        lmbind(LIGHT0, 1) ;
        lmbind(LMODEL, 1) ;
}
Osi()
{
float ce;
c3f(redvec);
ce =(maxs[2]-mins[2])/2;
move(0.0,0.0,ce);
draw(1.0,1.0,ce);

/*cmov (-0.6,0.499,ce);
charstr("X");*/
move(-0.5,0.5,ce);
draw(-0.5,0.6,ce);
/*cmov (-0.499,0.6,ce);
charstr("Y");*/
move(-0.5,0.5,ce);
draw(-0.5,0.5,ce-0.1);
/*cmov (-0.5,0.499,ce-0.1);
charstr("Z");*/



}

DO_JOB()
{
int mmm;
        mmm=80;
        c3f(backvec) ;
        clear() ;
        zclear() ;
for(;;)  {
        while (pnl_dopanel()) ;
        reshapeviewport();
        DrawObject();
        Osi();
        tumbl = tumble -> val;
        if(tumbl == 1)
        tum += mm ;
}
}



main(argc, argv)
 int argc ;
 char * argv[] ;
{
        int i, temp2;
        short  val ;
        int dev;

        if (strchr(argv[0], 'n'))
                negflag = 0 ;
        ReadObject() ;

        keepaspect(1, 1) ;
        foreground() ;
        winopen("surface") ;
        RGBmode() ;
        doublebuffer() ;
        gconfig() ;

        lsetdepth(0, 0x7FFFFF) ;
        zbuffer(TRUE) ;

        mmode(MVIEWING) ;

        DefPanel() ;
        pnl_needredraw();
        def_simple_light_calc();
DrawObject();
        add_event(winget(),REDRAW,ANY,DO_JOB(),0);
        qdevice (REDRAW);
        }

 float kvadrat1[] = {-1.0,-1.0,-1.0};
 float kvadrat2[] = {-1.0,1.0,-1.0};
 float kvadrat3[] = {1.0,1.0,-1.0};
 float kvadrat4[] = {1.0,-1.0,-1.0};

 Drawbox()
 {
 char string1[5],string2[5],string3[5],string4[5];

 sprintf(string1,"%s","X");
 sprintf(string2,"%s","Y");
 sprintf(string3,"%s","Z");
 sprintf(string4,"%s","O");
  c3f(whitevec);


 cmov(1.0,-0.97,-1.0);
 charstr(string1);
 cmov(-1.0,1.03,-1.0);
 charstr(string2);
 cmov(-1.03,-1.03,1.03);
 charstr(string3);
 cmov(-0.97,-0.97,-0.97);
 charstr(string4);
```

```
    pushmatrix();
    bgnclosedline();
    v3f(kvadrat1);
    v3f(kvadrat2);
    v3f(kvadrat3);
    v3f(kvadrat4);
    endclosedline();
    popmatrix();
    pushmatrix();
    translate(0.0,0.0,2.0);
    bgnclosedline();
    v3f(kvadrat1);
    v3f(kvadrat2);
    v3f(kvadrat3);
    v3f(kvadrat4);
    endclosedline();
    popmatrix();

    pushmatrix();
    rotate(-900,'y');
    bgnclosedline();
    v3f(kvadrat1);
    v3f(kvadrat2);
    v3f(kvadrat3);
    v3f(kvadrat4);
    endclosedline();
    popmatrix();
    pushmatrix();
    translate(-2.0,0.0,0.0);
    rotate(-900,'y');
    bgnclosedline();
    v3f(kvadrat1);
    v3f(kvadrat2);
    v3f(kvadrat3);
    v3f(kvadrat4);
    endclosedline();
    popmatrix();
    }
```

```
      subroutine adds(finam,madst,ind5,ind6)
c
c     tpiahgulqciq opophyx to~ek i dobablehie dualxhyx to~ek

      parameter(maxst=70000, maxtr=2*maxst)
      parameter(rvmin=1+1e-5)
      integer next,sites,tgl
c     'nst' - # to~ek, 'r' - ix koopdihaty
      common /points/ nst,r(4,maxst)
c     'ntr' - # tpeugolxhik ob, '/net/' - cpicok cmevhocti dlq
c     gpafa dualxhogo k tpiahgulqcii
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      common /cones/ vec(3,maxtr)
      common /bld/  radv(maxtr),iord(maxtr)
      common/extremum/rmin,rmax,radius,rad_ex
      logical ok
      logical frametr
      character*40 finam

1045    format(a40)
      open(1,file=finam)
      open(2,file='stempor',form='unformatted',status='unknown')
      open(3,file='saddual',form='unformatted',status='unknown')
      open(7,file='triang')
      open(8,file = 'text')

      nst=0
   10     continue
          nst=nst+1
          read (1,*,end=1020,err=1020) (r(i0,nst),i0=1,4)
      goto 10
 1020 nst=nst-1
        rmin = 1.e50
        rmax = -1.e50
        do 788 i = 1,nst
        rmin = min(rmin,r(4,i))
        rmax = max(rmax,r(4,i))
  788   continue

ccc     print *,'   ===>',nst,' points have been read    <===='
      if(nst.ge.maxst) then
         print *,'   ===>  too much for arrays been declared   <===='
         stop 12
         end if
      call tetra
c     ha~. kohfigupaciq - tetpaidp c bep(.hami 1,2,3,4
      tgl=1
      kkk=1000
      do 20 np=5,nst
         if(nst.ge.kkk.and.mod(np,kkk).eq.0) then
cc             print *,'   ===>',np,' points have been included  <===='
           end if
c        bkl~itx to~ku 'np' b tpiahgulqci'. dlq itogo
c        hajti tp-k 'tgl', b kotopom levit 'np'
         call search(tgl,np,ok,mk)
         if(ok) then
c            cdelatx bapicehtpi~eckoe podpazdelehie
             call bar(tgl,np,mk)
c            pepectpoitx tpiahgulqci'
             call recon(tgl)
             end if
   20    continue
c        print *,' triangulation is constructed'
      if(ind6.eq.1)then
      call graphics(igid)
      call output1
      end if

      if(ind6.eq.0)then
      call output
      end if

      if(nst .ge. madst)then
       write(8,*) '40 60'
       write(8,*) 'triangulation of inital points is constructed'
       write(8,*) 'the second input parameter is too small'
       write(8,*)'interpolation is not aloud'
       close(8)
       call try1("text")
       close(3)
       close(1)
       ind5 = 1
       if(ind6.eq.1) call winclo(igid)
       return
       end if

       ind5 = 0
       nst0=nst
       ic=0
 1040     ic=ic+1
          do 30 i2=1,ntr
              radv(i2)=scal(vec(1,i2),vec(1,i2))
   30         continue
          call kb07a(radv,ntr,iord)
ccc         print *,'max rad vec =',radv(ntr)
          it=ntr
 1050     if(nst.eq.madst.or.it.lt.2*ntr/3.or.
     .                    radv(it).lt.rvmin**2) goto 1060
              rv=sqrt(scal(vec(1,iord(it)),vec(1,iord(it))))
              if(rv.lt.rvmin) then
                  it=it-1
                  goto 1050
                  end if
              do 40 j1=1,3
                  r(j1,nst+1)=vec(j1,iord(it))/rv
   40             continue
              tgl=iord(it)
              call search(tgl,nst+1,ok,mk)
              if(ok.and..not.frametr(tgl,r,sites)) then
                  r(4,nst+1)=(r(4,sites(1,tgl))+r(4,sites(2,tgl))
     .            +r(4,sites(3,tgl)))/3
                  call bar(tgl,nst+1,mk)
                  call recon(tgl)
                  nst=nst+1
                  if(mod(nst-nst0,1000).eq.0) then
ccc                     print *,'   ===>',nst-nst0,
ccc     .                       ' dual points have been added  <===='
                  end if
              end if
              it=it-1
              if(ntr.gt.maxtr) then
                  print *,' maxtr is too small'
                  stop 13
                  end if
              goto 1050
 1060     continue
ccc         print *,'  # iterations =',ic
      if(it.lt.ntr.and.nst.lt.maxst) goto 1040
```

```
ccc     print *,'   ===>   ok! i write ...    <=== '
ccc     print *,'   ===>   # pnt =',nst,' <=== '
      write (3) ntr,nst
      write (3) ((r(i2,i3),i2=1,4),i3=1,nst)
      write (3) ((sites(i2,i3),i2=1,3),i3=1,ntr)
      close(3)
      if(ind6.eq.1) call winclo(igid)
      return
      end

      subroutine tetra
c
*     ihicializaciq dlq cfepi~eckoj tpiahgulqcii

      parameter(maxst=70000, maxtr=2*maxst)
      parameter(e=1e-6
     .         ,e1=0.1)
      integer next,sites,tgl
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      logical flag
      save

      if(nst.lt.4) then
          print *,'   ===>   data are wrong  <=== '
          stop 21
          end if

*     bektop 'r' dolveh imetx edihi~hu  ~.'hu
      flag=.false.
      do 10 n1=1,nst
         rad=scal(r(1,n1),r(1,n1))
         if(abs(rad-1).gt.e) then
            print *,' bektop r dolveh imetx dlihu = 1'
            print '(16,'' /r/ ='',g18.9)',n1,rad
            flag=.true.
            end if
         if(r(4,n1).le.0) then
            print *,' padluc-bektop dolveh bytx polovitelxhym'
            print '(''    r (4,'',i4,'') ='',g18.9)',n1,r(4,n1)
            flag=.true.
            end if
   10    continue
      if(flag) stop 22

*     ha~. kohfigupaciq ~ tetpaidp c bepihami 1,2 3,4
      ntr=4
      next(1,1)=4
      next(2,1)=2
      next(3,1)=3
      next(1,2)=4
      next(2,2)=3
      next(3,2)=1
      next(1,3)=4
      next(2,3)=1
      next(3,3)=2
      next(1,4)=1
      next(2,4)=3
      next(3,4)=2
      sites(1,1)=4
      sites(2,1)=2
      sites(3,1)=3
      sites(1,2)=4
      sites(2,2)=3
      sites(3,2)=1
      sites(1,3)=4
      sites(2,3)=1
      sites(3,3)=2
      sites(1,4)=1
      sites(2,4)=3
      sites(3,4)=2

*     ha~alo koopdihat dolvho levatx bhutpi tetpaidpa
 1010 do 20 n2=2,nst
         t1=scal(r(1,1),r(1,n2))
         if(t1.lt.-.2.and.t1.gt.-.8) goto 1020
   20    continue
      print *,'     ===>   initial data are wrong    <=== '
      print *,'     ===>            stop            <=== '
      stop 23
 1020 continue
      do 30 j0=1,4
         u1=r(j0,2)
         r(j0,2)=r(j0,n2)
         r(j0,n2)=u1
   30    continue
      do 40 n3=3,nst
         t2=volum(r(1,2),r(1,1),r(1,n3))
         if(abs(t2).gt..2) goto 1030
   40    continue
      print *,'     ===>   initial data are wrong    <=== '
      print *,'     ===>            stop            <=== '
      stop 24
 1030 continue
      do 50 j1=1,4
         u2=r(j1,3)
         r(j1,3)=r(j1,n3)
         r(j1,n3)=u2
   50    continue
      if(t2.lt.0) then
         do 60 j2=1,4
            u3=r(j2,3)
            r(j2,3)=r(j2,1)
            r(j2,1)=u3
   60       continue
         end if
      do 70 n4=4,nst
         v1=volum(r(1,2),r(1,3),r(1,n4))
         if(v1.gt.e1) then
            v2=volum(r(1,3),r(1,1),r(1,n4))
            if(v2.gt.e1) then
               v3=volum(r(1,1),r(1,2),r(1,n4))
               if(v3.gt.e1) goto 1040
               end if
            end if
   70    continue
      print *,'     ===>   initial data are wrong    <=== '
      print *,'     ===>            stop            <=== '
      stop 25
 1040 continue
      do 80 j2=1,4
         u3=r(j2,4)
         r(j2,4)=r(j2,n4)
         r(j2,n4)=u3
   80    continue
      call circle(1)
      call circle(2)
      call circle(3)
```

```
      call circle(4)
      return
      end

      subroutine bar(tr1,st,mk)
*     _____________________
*     to-ka 'st' pome}aetcq b tp-k 'tr1' ppi pomo}i
*     bapicehtpl~eckogo podpazdelehiq

      parameter(maxst=70000, maxtr=2*maxst)
      integer next,sites,np,tr1,tr2,tr3,tra,trb,st,st1,st2,st3
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      save

      if(mk.ne.0) then
         n0=next(mk,tr1)
         m0=mark(n0,tr1)
         endif
      tr2=next(2,tr1)
      tr3=next(3,tr1)
      m2=mark(tr2,tr1)
      m3=mark(tr3,tr1)
*     'tra' i 'trb' - dba hobyx tpeugolxhika
      tra=ntr+1
      trb=ntr+2
      ntr=ntr+2
      next(m2,tr2)=tra
      next(m3,tr3)=trb
      next(2,tr1)=tra
      next(3,tr1)=trb
      next(1,tra)=tr2
      next(2,tra)=trb
      next(3,tra)=tr1
      next(1,trb)=tr3
      next(2,trb)=tr1
      next(3,trb)=tra
      st1=sites(1,tr1)
      st2=sites(2,tr1)
      st3=sites(3,tr1)
      sites(1,tr1)=st
      sites(1,tra)=st
      sites(2,tra)=st3
      sites(3,tra)=st1
      sites(1,trb)=st
      sites(2,trb)=st1
      sites(3,trb)=st2
      if(mk.ne.0) then
         n1=next(m0,n0)
         call flip(n1,1)
         call circle(n0)
         endif
      call circle(tr1)
      call circle(tra)
      call circle(trb)
      return
      end

      subroutine search(tgl,np,ok,mk)
*     _______________________________
*     poick tpeugolxnika b kotopyj popadaet to-ka np

      parameter(maxst=70000, maxtr=2*maxst)
      parameter (e=1e-24
     .          ,el=1-1e-6)
      integer next,sites,tgl,m,np,prev,ndp/0/
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      logical ok
      save

      mk=0
      a3=volum(r(1,np),r(1,sites(1,tgl)),r(1,sites(2,tgl)))
      a1=volum(r(1,np),r(1,sites(2,tgl)),r(1,sites(3,tgl)))
      a2=volum(r(1,np),r(1,sites(3,tgl)),r(1,sites(1,tgl)))
      if(a1.ge.0.and.a2.ge.0.and.a3.ge.0) then
          v1=abs(a1)
          v2=abs(a2)
          v3=abs(a3)
          if(v1.le.e.or.v2.le.e.or.v3.le.e) then
              if(v1.le.e.and.v2.le.e.and.v3.le.e) then
                  print *,'  all volumes are zero'
                  endif
              if(v1.le.e) then
                  if(v2.le.e) then
                      ok=.false.
                      ndp=ndp+1
                      print *,ndp,'-th point is deleted'
                      if(scal(r(1,np),r(1,sites(3,tgl))).ge.el) then
                          r(4,sites(3,tgl))=(r(4,np)
     .                     +r(4,sites(3,tgl)))/2
                          end if
                  else
                      if(v3.le.e) then
                          if(scal(r(1,np),r(1,sites(2,tgl))).ge.el) then
                              r(4,sites(2,tgl))=(r(4,np)
     .                         +r(4,sites(2,tgl)))/2
                              end if
                          ok=.false.
                          ndp=ndp+1
                          print *,ndp,'-th point is deleted'
                      else
                         mk=1
                         ok=.true.
                         end if
                      endif
              else
                  if(v2.gt.e) then
                      mk=3
                      ok=.true.
                  else
                      if(v3.le.e) then
                          if(scal(r(1,np),r(1,sites(1,tgl))).ge.el) then
                              r(4,sites(1,tgl))=(r(4,np)
     .                         +r(4,sites(1,tgl)))/2
                              end if
                          ok=.false.
                          ndp=ndp+1
                          print *,ndp,'-th point is deleted'
                      else
                         mk=2
                         ok=.true.
                         endif
                      endif
                  endif
          else
              ok=.true.
              end if
          return
```

```
          end if
        if(a2.lt.a3) then
            if(a1.lt.a2) then
                m=1
                a=a1
            else
                m=2
                a=a2
                end if
        else
            if(a1.lt.a3) then
                m=1
                a=a1
            else
                m=3
                a=a3
                end if
            end if

        k0=0
1010    prev=tgl
        tgl=next(m,tgl)
        m2=mark(tgl,prev)
        m3=mod(m2,3)+1
        m1=mod(m3,3)+1
        a3=volum(r(1,np),r(1,sites(m1,tgl)),r(1,sites(m2,tgl)))
        a1=volum(r(1,np),r(1,sites(m2,tgl)),r(1,sites(m3,tgl)))
        a2=-a
        if(a1.lt.a3) then
            m=m1
            a=a1
        else
            m=m3
            a=a3
            end if
        k0=k0+1
        if(k0.gt.maxtr) then
            print *,' ===> tpeugolxhik he hajdeh <===='
            ok=.false.
            return
            end if
        if(a.lt.0) goto 1010
        v1=abs(a1)
        v2=abs(a2)
        v3=abs(a3)
        if(v1.le.e.or.v2.le.e.or.v3.le.e) then
            if(v1.le.e.and.v2.le.e.and.v3.le.e) then
                print *,'  all volumes are zero'
                endif
            if(v1.le.e) then
                if(v2.le.e) then
                    if(scal(r(1,np),r(1,sites(m3,tgl))).ge.e1) then
                        r(4,sites(m3,tgl))=(r(4,np)
     .                   +r(4,sites(m3,tgl)))/2
                        end if
                    ok=.false.
                    ndp=ndp+1
                    print *,ndp,'-th point is deleted'
                else
                    if(v3.le.e) then
                        if(scal(r(1,np),r(1,sites(m2,tgl))).ge.e1) then
                            r(4,sites(m2,tgl))=(r(4,np)
     .                       +r(4,sites(m2,tgl)))/2
                            end if
                        ok=.false.
                        ndp=ndp+1
                        print *,ndp,'-th point is deleted'
                    else
                        mk=m1
                        ok=.true.
                        end if
                    end if
            else
                if(v2.gt.e) then
                    mk=m3
                    ok=.true.
                else
                    if(v3.le.e) then
                        if(scal(r(1,np),r(1,sites(m1,tgl))).ge.e1) then
                            r(4,sites(m1,tgl))=(r(4,np)
     .                       +r(4,sites(m1,tgl)))/2
                            end if
                        ok=.false.
                        ndp=ndp+1
                        print *,ndp,'-th point is deleted'
                    else
                        mk=m2
                        ok=.true.
                        endif
                    endif
            endif
        else
            ok=.true.
            mk=0
            end if
        return
        end

        subroutine recon(tgl)
*       ---------------------
*       pepectpojka pe[etki.
*       bocctahoblehie tpiahgulqcii delohe
*       ppi pomo}i kpugobogo kpitepiq

        parameter(maxst=70000, maxtr=2*maxst)
        integer next,sites,tgl,nxt,ngb,stp
        common /points/ nst,r(4,maxst)
        common /net/ ntr,next(3,maxtr),sites(3,maxtr)
        logical criter
        save

        nxt=tgl
        stp=0
1010    ngb=next(1,nxt)
        if(sites(1,nxt).ne.sites(1,tgl)) then
            print *,'   ===>   next i sites he coglacobahy   <===='
            stop 41
            end if
        if(.not.criter(ngb,sites(1,tgl))) then
            call flip(nxt,1)
            goto 1010
            end if
        call circle(nxt)
        nxt=next(2,nxt)
        stp=stp+1
        if(stp.gt.3*nst)  then
            print *,'    ===>   tpiahgulqciq delohe he polut}aetcq   <===='
            return
```

```
      end if
      if(nxt.ne.tgl) goto 1010
      return
      end

      logical function criter(tg,st)
c     ------------------------------
c     ppobepka kpugobogo kpitepiq:
c     popadaet li to~ka st b kohuc, opicahhyj bokpug tp-ka tg ?
c     ecli popadaet, to criter = .false.

      parameter(maxst=70000, maxtr=2*maxst)
      integer next,sites,tg,st
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      common /cones/ vec(3,maxtr)
      save

      if(scal(r(1,st),vec(1,tg)).gt.1) then
          criter=.false.
      else
          criter=.true.
          end if
      return
      end

      subroutine flip(tga,mal)
c     ------------------------

      parameter(maxst=70000, maxtr=2*maxst)
      integer next,sites,tga,tgb,shl
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      save
      shl(m)=mod(m,3)+1

      ma2=shl(mal)
      ma3=shl(ma2)
      tgb=next(mal,tga)
      mb1=mark(tgb,tga)
      mb2=shl(mb1)
      mb3=shl(mb2)
      next(mark(next(ma2,tga),tga),next(ma2,tga))=tgb
      next(mark(next(mb2,tgb),tgb),next(mb2,tgb))=tga
      next(mal,tga)=next(mb2,tgb)
      next(mal,tgb)=next(mb3,tgb)
      next(ma2,tgb)=next(ma2,tga)
      next(ma2,tga)=tgb
      next(ma3,tgb)=tga
      sites(ma3,tga)=sites(mb1,tgb)
      sites(ma3,tgb)=sites(mb2,tgb)
      sites(mal,tgb)=sites(mal,tga)
      sites(ma2,tgb)=sites(ma3,tga)
      return
      end

      subroutine circle(tgl)
c     ----------------------
c     by~iclq'tcq kompohehty bektopa, happablehhogo bdolx
c     oci opicahhogo bokpug tp~ka tgl kohuca l
c     ime'}ego taku' dlihu, ~to ego ckalqphoe ppoizbedehie
c     c bektopami obpazu'}ix kohuca pabho 1.

      parameter(maxst=70000, maxtr=2*maxst)
      integer next,sites,tgl
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      common /cones/ vec(3,maxtr)
      real a(3,3),rhs(3),iw(3),w(6),x(3)
      ind(i)=mod(i,3)+1
      save

      do 10 i0=1,3
          rhs(i0)=1
          do 10 i1=1,3
   10         a(i0,i1)=r(i1,sites(i0,tgl))
      call ma14a(3,3,0,a,3,rhs,x,iw,w,+1)
      do 20 j1=1,3
   20     vec(j1,tgl)=x(j1)
      return
      end

      real function scal(x,y)
c     -----------------------
c     scal = (x,y)
      real x(3),y(3)

      scal=dprod(x(1),y(1))+dprod(x(2),y(2))+dprod(x(3),y(3))
      return
      end

      real function volum(x,y,z)
c     --------------------------
c     ob'em co zhakom
c     volume = (x,[y,z])
      real x(3),y(3),z(3)
      real*8 vol

      vol=0
      do 10 j1=1,3
          m=mod(j1,3)+1
          n=mod(j1+1,3)+1
          vol=vol+x(j1)*(dprod(y(m),z(n))-dprod(y(n),z(m)))
   10     continue
      volum=vol
      return
      end

      integer function mark(tr1,tr2)
c     ------------------------------
      integer tr1,tr2

      parameter(maxst=70000, maxtr=2*maxst)
      integer next,sites
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)

c     oppedelehie opiehtacii b tp-ke tr1 po tp-ku tr2
      do 10 mark=1,3
          if(next(mark,tr1).eq.tr2) return
   10     continue
      print *,'     --->   cpicok cmevhoctej he camocoglacobah   <---'
      stop 51
      end

      subroutine output
c     bybod dlq bizualizacii i ihteppolqcii

      parameter(maxst=70000, maxtr=2*maxst)
```

```
      integer next,sites
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      logical addpts
      save
      addpts=.true.

      write (7,*) ntr,nst
      do 10 j2=1,nst
          write (7,*) ((r(i1,j2)*r(4,j2)),i1=1,3)
   10     continue
      do 11 j2=1,ntr
          write (7,*) sites(1,j2),sites(2,j2),sites(3,j2)
   11     continue
      close(7)
      if(.not addpts) stop
      write (2) ntr,nst
      write (2) ((sites(i0,k0),i0=1,3),k0=1,ntr)
      write (2) ((next(i0,k0),i0=1,3),k0=1,ntr)
      close(2)
      close(1)
      return
      end
```

```
      subroutine output1
      parameter(maxst=70000, maxtr=2*maxst)
      integer next,sites
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      common/extremum/rmin,rmax,radius,rad_ex
      common /bld/  radv(maxtr),iord(maxtr)
      logical addpts
      logical frametr
      save
      addpts=.true.

      ntr7 = 0

      do  70 i = 1,ntr
      if(frametr(i,r,sites)) goto 70
      ntr7 = ntr7 + 1
      iord(ntr7) = i
   70 continue
      write (7,*) ntr7,nst
      do 71 j2=1,nst
          write (7,*) ((r(i1,j2)*r(4,j2)),i1=1,3)
   71     continue
      do 72 i3=1,ntr7
          j2 = iord(i3)
          write (7,*) sites(1,j2),sites(2,j2),sites(3,j2)
   72     continue
      close(7)
      if(.not.addpts) stop
      write (2) ntr,nst
      write (2) ((sites(i0,k0),i0=1,3),k0=1,ntr)
      write (2) ((next(i0,k0),i0=1,3),k0=1,ntr)
      close(2)
      close(1)
      return
      end
```

```
      logical function frametr( nt,r,sites )
        real r(4,1)
        integer sites(3,1)

      frametr=.false.
      do 1 k=1,3
      frametr=frametr.or.(r(4,sites(k,nt)).ge. 1.5-(1.d-5))
    1  continue
      return
        end
```

```
      subroutine add(msize)
*
*     ____________________
*     adding of dual points
*
      parameter(maxst=70000
     .         ,mmaxtr=2*maxst)
      common /points/ nst,ntr
      real r(3,maxst)
      integer next(3,mmaxtr),sites(3,mmaxtr)
      common/lala/next,sites,r
      real radius(mmaxtr),center(2,mmaxtr)
      real radv(mmaxtr),iord(mmaxtr)
      common /work1/ radius,center, radv,iord
        common /iider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,iig,miner,kva
     &              ,ipik,ipog

       open(3,file='addual',status='unknown',form='unformatted')
       open(8,file='text')
       call glider
       write(8,*)'40 30'
       write(8,*)' adding of dual points:'
       write(8,15) 'the number of the points before adding is ',nst
  15   format(a,i7)
*       *    adding of dual points
       if(madst.gt.maxst) then
          print *,'===> dimension maxst is too small <==='
          print *,'===>     put     maxst    = ',madst,' <==='
          stop
          end if
       madst1 = nst*msize
       call addual(madst1,rdmax,r,next,sites,radius,center,radv,iord)
       write(8,15)'the number of the points after adding is ',nst

      ntr7=0
      do 3777 nt1=1,ntr
c     if(radius(nt1).ge.rdmax) goto 3777
      ntr7=ntr7+1
      iord(ntr7)=nt1
 3777 continue
      write (3) ntr7,nst
      write (3) ((r(i2,i3),i2=1,3),i3=1,nst)
      write (3) ((sites(i2,int(iord(i3)) ),i2=1,3),i3=1,ntr7)
      call plide('addual')
c     print *,'   ===>',ntr7,'triangles    <=== '
      idr1=idra
      if(idra.lt.0) then
         print *, 'That is the trap'
c        print *, 'draw the new triangulation ?'
c             call getans(idr1)
         endif
*     if(idr1.ne.0) call dtri(150,r,sites)
c      print *,'   ===>   the job is done   <=== '
      close(3)
      write(8,*)'    Hit right mouse button to continue'
      close(8)
      return
      end


      subroutine addual(madst,rdmax,
     &r,next,sites,radius,center,radv,iord)
      parameter(rvmin=1e-8,rvmax=0.7)
      common /points/ nst,ntr
      real r(3,1)
      integer next(3,1),sites(3,1)
      real radius(1),center(2,1)
      real radv(1)
      integer iord(1)
      integer tgl,temp
      logical ok,frame,bad,border
*
      if(madst.le.nst) return

 2000   continue
      rav=rvmin
       rma=rav
       nst0=nst

  888   nc=0
       ral=0d0
       naver=0
       aver=0
       tgl=1

       do 789 i=1,ntr
       if(bad(i,rav,radius,circle)) goto 789
            r(1,nst+1)=center(1,i)
             r(2,nst+1)=center(2,i)
             r(3,nst+1)=0
             call search(tgl,nst+1,ok,mk,r,next,sites)
               if(.not.ok) goto 789
      if(border(tgl,r,sites,radius)) goto 789
      naver=naver+1
       aver=aver+radius(i)
       ral=max(ral,radius(i))
  789  continue
              if(naver.eq.0) then
                      print *,' can not add points'
                      write(8,*) 'can not add points'
                      return
                      endif
       rma=aver/(1.01*naver)
 5678  continue

       do 24 i=1,ntr
       if(bad(i,rma,radius,circle)) goto 24
       nss=1+nc+nst
             r(1,nss)=center(1,i)
             r(2,nss)=center(2,i)
             r(3,nss)=0
          call search(tgl,nss,ok,mk,r,next,sites)
             if(.not.ok) goto 24
          if(border(tgl,r,sites,radius)) goto 24
             do 644 k=1,3
             nt=next(k,i)
             if(nt.gt.i) goto 644
             if(bad(nt,rma,radius,circle)) goto 644
           r(1,nss+1)=center(1,nt)
           r(2,nss+1)=center(2,nt)
           ngl=tgl
          call search(ngl,nss+1,ok,mk,r,next,sites)
             if(.not.ok) goto 644
          if(border(ngl,r,sites,radius)) goto 644
             ra=(center(1,nt)-center(1,i))**2
             ra=(center(2,nt)-center(2,i))**2+ra
             if(ra.lt.rav) goto 24
  644   continue
```

```
                nc=nc+1
                if(nss.ge madst) goto 103
24        continue
10?       continue
                if(nc.le.1) then
c               print *,' can not add  more points',rma,rav
           write(8.``'can not add more points'
                        return
                        endif
          nss=nc+nst

                do 890 ms=nst0+1,nss
                call search(tgl,ms,ok,mk,r,next,sites)
                  if(.not.ok) goto 890
                nst=nst+1
                r(1,nst)=r(1,ms)
                r(2,nst)=r(2,ms)
                call bar(tgl,nst,mk,r,next,sites,radius,center)
                call recon(tgl,r,next,sites,radius,center)
 890             continue
c               print *,'    ===>',nst-nst0,' out of ',madst-nst0,
c     .                           ' dual points have been added  <===='
        if(madst.gt.nst) goto 2000

c1070 print *,
c     & ' ===>  the  addition of dual points is completed <===='
C       print *,'    ===>    # pnt =',nst,' <=== '
        return
        end
```

```
      subroutine graphics(igid)
#include<fgl.h>
#include<fdevice.h>
      parameter(maxst=70000, maxtr=2*maxst)
      integer next,sites
       common /points/ nost,r(4,maxst)
        common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      common/mode/ exitmode,pictmode,pointmode
      common/extremum/rmin,rmax,radius,rad_ex
      integer exitmode,pictmode,pointmode

      call init(igid)
      call clear
      call zclear
      call swapbu
      do while(1)
      do while(qtest(0).eq.0)
      call Draw
      end do
      call process
        if(exitmode .eq.0)goto 5
      end do
  5   return
      end



      subroutine init(igid)
#include<fgl.h>
#include<fdevice.h>
      parameter(NORMAL = 1,HOLE = 0)
      parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
      common/mode/ exitmode,pictmode,pointmode
      common/extremum/rmin,rmax,radius,rad_ex
       common/shtri/ sh(4),co(3),con(3)
      common/rots/ angl(7),pov(3)
      common/col/ color,mcolor,legend(3,13)
      common/popup_stuff/menu
      integer exitmode,pictmode,pointmode
       real backvec(3)/0.0,0.0,0.0/
      call prefpo(0,1270,0,1000)
      call foregr
      igid = winope("geosphere",13)
      call RGBmod
      call double
      call gconfi
      call lsetde (0, $7fffff)
      call zbuffe(.TRUE.)
      call qdevic(RIGHTM)
      call qdevic(LEFTMO)
      call tie(RIGHTM,MOUSEX,MOUSEY)
      call tie(LEFTMO,MOUSEX,MOUSEY)
      menu = newpup()
      call addtop(menu,'DATA CHANGE %t|WIRED MODEL|MAP|SMOOTH MAP',50,0)
      call addtop(menu,' add point|remove point|EXIT  ',50,0)
      exitmode = 1
      pictmode= 1
      pointmode = NORMAL
      do 12 i = 2,4
12    sh(i) = x1+0.001
      do 16 i = 5,7
16    angl(i) = x1
      call c3f(backvec)
      color = x1+w
      mcolor = 12
      rad_ex = x1+w/8 +rmin*w*0.75
      return
      end

      subroutine process
#include<fgl.h>
#include<fdevice.h>
      common/mode/ exitmode,pictmode,pointmode
      common/popup_stuff/menu
      common/extremum/rmin,rmax,radius,rad_ex
      integer exitmode,pictmode,pointmode
      integer*2 val,mx,my
      integer dev
      integer menuval,exitmode,pictmode,pointmode
      dev = qtest()
      do 1000 while (dev .ne. 0)
      dev = qread(val)
      if(dev .eq. LEFTMO) then
      if(val .eq. 1) then
      i = qread(mx)
      k = qread(my)
       call move_shtrih(mx,my)
       call Draw
      do 1001 while (qtest() .eq.0)
      call Draw
1001  end do
      end if
      else
      if(dev.eq.RIGHTM) then
      menuval = dopup(menu)
       if(menuval .eq.1)then
        pictmode= 1
        call Draw
        end if
        if(menuval .eq.2)then
        pictmode= 2
        call Draw
        end if
        if(menuval .eq.3)then
        pictmode= 3
        call Draw
        end if
         if(menuval .eq.4)then
         call add_point
        call Draw
         end if
         if(menuval .eq.5)then
         call remove_point
        call Draw
         end if
        if(menuval.eq.6) then
        exitmode =  0
        if(exitmode .eq.0)goto 6
        end if
      end if
      end if
1000  end do
6     return
      end
```

```
      subroutine Draw
#include<fgl.h>
#include<fdevice.h>
       common/shtri/ sh(4),co(3),con(3)
      common/rots/ angl(7),pov(3)
      common/col/ color,mcolor,legend(3,13)
      common/extremum/rmin,rmax,radius,rad_ex
      character string1*5,string2*5,string3*5,string4*5
       real backvec(3)/0.0,0.0,0.0/
       real greenvec(3)/1.0,1.0,1.0/

       real idmat(4,4)/
     & 0.5,0.0,0.0,0.0,
     & 0.0,0.5,0.0,0.0.
     & 0.0,0.0,0.5,0.0,
     & 0.0,0.0,0.0,0.5/

      real kvadrat1(3)/-1.0,-1.0,-1.0/
      real kvadrat2(3)/-1.0,1.0,-1.0/
      real kvadrat3(3)/1.0,1.0,-1.0/
      real kvadrat4(3)/1.0,-1.0,-1.0/


      write(string1,*)'X'
      write(string2,*)'Y'
      write(string3,*)'Z'
      write(string4,*)'O'
      call c3f(backvec)
      call clear
      call zclear

      call ortho(-1.5,2.1,-1.5,1.5,-1.5,1.5)
      call loadma(idmat)
      call palette(idmat)
      call pushma
      call rot(pov(5,,'x')
      call rot(pov(6),'y')
      call rot(pov(7),'z')
      call c3f(greenvec)
      call cmov( 1.03,-1.03,-1.03)
      call charst(string1,5)
      call cmov(-1.03,1.03,-1.03)
      call charst(string2,5)
      call cmov(-1.03,-1.03,1.03)
      call charst(string3,5)
      call cmov(-1.03,-1.03,-1.03)
      call charst(string4,5)
      call c3f(greenvec)
      call bgnclo
      call v3f(kvadrat1)
      call v3f(kvadrat2)
      call v3f(kvadrat3)
      call v3f(kvadrat4)
      call endclo

      call pushma
      call transl(0.0,0.0,2.0)
      call c3f(greenvec)
      call bgnclo
      call v3f(kvadrat1)
      call v3f(kvadrat2)
      call v3f(kvadrat3)
      call v3f(kvadrat4)
      call endclo
      call popmat
      call pushma
      call rotate(-900,'y')
      call c3f(greenvec)
      call bgnclo
      call v3f(kvadrat1)
      call v3f(kvadrat2)
      call v3f(kvadrat3)
      call v3f(kvadrat4)
      call endclo
      call popmat
      call pushma
      call transl(-2.0,0.0,0.0)
      call rotate(-900,'y')
      call c3f(greenvec)
      call bgnclo
      call v3f(kvadrat1)
      call v3f(kvadrat2)
      call v3f(kvadrat3)
      call v3f(kvadrat4)
      call endclo
       call popmat
       call Obj
      call make_metka
       call popmat
      call swapbu
      return
      end



      subroutine palette(idmat)
#include<fgl.h>
#include<fdevice.h>
      common/shtri/ sh(4),co(3),con(3)
      common/rots/ angl(7),pov(3)
      common/col/ color,mcolor,legend(3,13)
      common/extremum/rmin,rmax,radius,rad_ex
      parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
      parameter(dl = 0.005)
      real legend(3,13)/
     &0.0,0.0,0.8,
     &0.0,0.0,1.0,
     &0.0,0.6,0.6,
     &0.0,0.8,0.0,
     &0.0,1.0,0.0,
     &0.2,0.9,0.0,
     &1.0,1.0,0.0,
     &1.0,0.8,0.0,
     &1.0,0.6,0.0,
     &1.0,0.4,0.0,
     &1.0,0.2,0.0,
     &1.0,0.0,0.0,
     &0.0,0.0,0.0/

      real bluevec(3)/0.0,0.8,1.0/
      real redvec(3)/1.0,0.0,0.0/
      real bluevec1(3)/0.0,0.0,1.0/
      real greyvec(3)/0.8,0.8,0.8/
      real blackvec(3)/0.0,0.0,0.0/
```

```
      character*40 string(8)

      write(string(1),*)'rmin  radius   rmax'
      write(string(2),*)'0   X coordinate  1'
      write(string(3),*)'0   Y coordinate  1'
      write(string(4),*)'0   Z coordinate  1'
      write(string(5),*)'0   X rotation  360'
      write(st      ),*)'0   Y rotation  360'
      write(st      ),*)'0   Z rotation  360'
      write(stri    8),*)'Hole rmin   rmax   1.5'
      call c3f(blackvec)
      call rectf(1.5,-1.5,2.1,1.5)
      call c3f(greyvec)
      call rectf(1.5,-1.5,2.1,1.5)
      do 20 i = 1,7
      x = x1
      y = dist*i + h*(i-1)+y1
      call c3f(blackvec)
      call rectf(x-d1,y-d1,x+w+d1,y+h+d1)
      call c3f(bluevec)
      call rectf(x,y,x+w,y+h)
      call c3f(blackvec)
      call cmov(x1,y-0.04,0.1)
      call charst(string(i),30)
20    continue
      do 21 k = 2,7
      y2= dist*k + h*(k-1)+y1
      do 23 j = 1,11
      if(mod(j,6) .eq.0)then
      h1 = 0.03
      else
      if(mod(j,3) .eq.0)then
      h1 = 0.02
      else
      h1 = 0.01
      end if
      end if
      w1 = 0.002
      x2= x1+w/12.*j
      call c3f(blackvec)
      call rectf(x2,y2,x2+w1,y2+h1)
23    continue
21    continue
      do 22  j = 0,11
      y = dist +y1
      x = x1+w/12.*j
      call c3f(legend(1,j+1))
      call rectf(x,y,x+w/12.,y+h)
22    continue
      y = y1
      x = x1
      w1 = 0.002
      h1 = 0.03
      w2 = 0.004
      call c3f(blackvec)
      call rectf(x-d1,y-h-d1,x+w+d1,y+d1)
      call c3f(bluevec)
      call rectf(x,y-h,x+w,y)
      call c3f(blackvec)
      x = x1
      call rectf(x,y-h,x+w/8,y)
      call c3f(blackvec)
      call rectf(x,y-h,x+w2,y-h+h1)
      x5 = x1 + w/8 +rmin*w*0.875/1.5
      call c3f(bluevec1)
      call rectf(x5,y-h,x5+w1,y)
      x5 = x1 + +w/8+rmax*w*0.875/1.5
      call c3f(redvec)
      call rectf(x5,y-h,x5+w1,y)
      call c3f(blackvec)
      call cmov(x1-0.07,y-h-0.04,0.1)
      call charst(string(8),30)
      call shtrih
      return
      end


      subroutine shtrih
      common/shtri/ sh(4),co(3),con(3)
      common/rots/ angl(7),pov(3)
      common/col/ color,mcolor,legend(3,13)
      common/extremum/rmin,rmax,radius,rad_ex
      parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
      parameter(del = 0.005)
      real blackvec(3)/0.0,0.0,0.0/
      i = 0
      y0 = y1-h
      call c3f(blackvec)
      call rectf(rad_ex,y0,rad_ex+del,y0+h)
      i = 1
      y0 = y1 + dist*i + h*(i-1)
      call c3f(blackvec)
      call rectf(color,y0,color+del,y0+h)
      do 11 i = 2,4
      y0 = y1 + dist*i + h*(i-1)
      call c3f(blackvec)
      call rectf(sh(i),y0,sn(i)+del,y0+h)
11    continue
      do 15 i = 5,7
      y0 = y1 + dist*i + h*(i-1)
      call c3f(blackvec)
      call rectf(angl(i),y0,angl(i)+del,y0+h)
15    continue
      return
      end


      subroutine move_shtrih(mx,my)
      parameter(NORMAL = 1,HOLE = 0)
      parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
      parameter(del = 0.005)
      common/shtri/ sh(4),co(3),con(3)
      common/rots/ angl(7),pov(3)
      common/col/ color,mcolor,legend(3,13)
        common/mode/exitmode,pictmode,pointmode
      common/extremum/rmin,rmax,radius,rad_ex
      integer*2 mx,my
      integer exitmode,pictmode,pointmode
      real whitevec(3)/1.0,1.0,1.0/
      real x3(13),x4(6)

*     Metky na shkale radiusov
      x4(1) = x1
      x4(2) = x1+w/8
      x4(3) = x4(2) + rmin * 0.875 *w/1.5
      x4(4) = x4(2) + rmax * 0.8888888875 *w/1.5
      x4(5) = x4(2) + 0.875* w/1.5
      x4(6) = x1+w
```

```
      x = (3.6/1270)*float(mx) - 1.5
      y = (3.0/1000)*float(my) - 1.5



      if(y.le.(y1-h).or. x.lt.x1 .or.x .gt. x1+w ) then
      n = -1
      else
       n = int((y - y1)/(dist+h))+1
       if (y - (y1 + dist*n +h*(n-1)) .le.0) n = -1
        if(y .le.y1 .and. y .gt.y1-h) n=0
       if(n.eq.0; then
       do 20 i = 1,6
       if((abs(x-x4(i))).le.2*del)then
       x = x4(i)
       goto 19
       end if
 20    continue
 19    rad_ex = x
       call make_change(x4)
       end if
       if(n.eq.1) then
       color = x
        call make_change_back
       end if
       if(n.le.4.and.n.ge.2)then
       sh(n) = x
       end if
       if(n.ge.5 .and.n.le.7)then
       do 17 i = 0,12
        x3(i) = x1+(w/12)*i
       if((abs(x-x3(i))).le.del)then
       x = x3(i)
       goto 18
       end if
 17    continue
 18    angl(n) = x
       end if
       end if
       do 14 i = 5,7
       pov(i) = ((angl(i) - x1)/w)*360
 14    continue
       if(pointmode .eq.NORMAL) then
       radius = ((color - x1)/w)*(rmax-rmin)+rmin
       mcolor = ((color - x1)/w)*11.+1
       end if
      co(1) = ((sh(2) - x1)/w)*2. - 1.
      co(2) = ((sh(3) - x1)/w)*2. - 1.
      co(3) = ((sh(4) - x1)/w)*2. - 1.
       x = co(1)
       y = co(2)
       z = co(3)
*******calc of the position of the normal.vec *
       s =sqrt( 1.0/(1.e-10+x**2+y**2+z**2))
       con(1) = s*x
       con(2) = s*y
       con(3) = s*z
******calc of the position of the point on the beam*
       co(1) = con(1)*radius
       co(2) = con(2)*radius
       co(3) = con(3)*radius
       return
       end

       subroutine make_metka
#include<fgl.h>
#include<fdevice.h>
      parameter(NORMAL = 1,HOLE = 0)
      parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
      parameter(piram=.015)
      common/shtri/ sh(4),co(3),con(3)
      common/mode/exitmode,pictmode,pointmode
      common/rots/ angl(7),pov(3)
      common/col/ color,mcolor,legend(3,13)
      common/extremum/rmin,rmax,radius,rad_ex
      integer exitmode,pictmode,pointmode
      real metka1(3),metka2(3),metka3(3),metka4(3),metka5(3)
      real metka6(3),metka7(3)
      real beam(3)
      integer*4 n
       real greenvec(3)/1.0,1.0,1.0/
       real zero(3)/0.0,0.0,0.0/
*******calc of the position of the beam
       radius = ((color - x1)/w)*(rmax-rmin)+rmin
       s1 =sqrt( 10.0/(1.e-10+x**2+y**2+z**2))
       beam(1) = s1*con(1)
       beam(2) = s1*con(2)
       beam(3) = s1*con(3)
******calc of the metka  on the beam*
       metka1(1) = -con(2)*piram+radius*con(1)
       metka1(2) = con(1)*piram+radius*con(2)
       metka1(3) = radius*con(3)
       metka2(1) = con(2)*piram+radius*con(1)
       metka2(2) = -con(1)*piram+radius*con(2)
       metka2(3) = radius*con(3)

       metka3(1) = radius*con(1)
       metka3(2) = -con(3)*piram+radius*con(2)
       metka3(3) = con(2)*piram+radius*con(3)
       metka4(1) = radius*con(1)
       metka4(2) = con(3)*piram+radius*con(2)
       metka4(3) = -con(2)*piram+radius*con(3)

       metka5(1) = -con(3)*piram+radius*con(1)
       metka5(2) = radius*con(2)
       metka5(3) = con(1)*piram+radius*con(3)
       metka6(1) = con(3)*piram+radius*con(1)
       metka6(2) = radius*con(2)
       metka6(3) = -con(1)*piram+radius*con(3)

       metka7(1) = (4*piram+radius)*con(1)
       metka7(2) = (4*piram+radius)*con(2)
       metka7(3) = (4*piram+radius)*con(3)
       if(pointmode .eq. NORMAL) then
****** metka*
       call c3f(greenvec)
       call bgnpol
       call v3f(metka1)
       call v3f(metka2)
       call v3f(metka7)
       call v3f(metka1)
       call endpol
       call bgnpol
       call v3f(metka3)
       call v3f(metka4)
```

```
      call v3f(metka7)
      call v3f(metka3)
      call endpol
      call bgnpol
      call v3f(metka5)
      call v3f(metka6)
      call v3f(metka7)
      call v3f(metka5)
      call endpol
      end if
****** beam*
      call c3f(greenvec)
      call bgnlin
      call v3f(zero)
      call v3f(beam)
      call endlin
      return
      end


      subroutine Obj
#include<fgl.h>
#include<fdevice.h>
      parameter(maxst=70000, maxtr=2*maxst)
       common/shtrl/ sh(4),co(3),con(3)
      common/rots/ angl(7),pov(3)
      common/col/ color,mcolor,legend(3,13)
      common/extremum/rmin,rmax,radius,rad_ex
      integer next,sites
      integer pictmode,pointmode
      real bordovec(3)/0.0,0.0,0.0/
       common /points/ nost,r(4,maxst)
        common /net/ ntr,next(3,maxtr),sites(3,maxtr)
        common/mode/exitmode,pictmode,pointmode
        real idmat(4,4)
        real x(3),y1(3),y2(3)
        integer z1
***************** MAP MODEL **********
        if(pictmode.eq.2) then
        do 30 i = 1,ntr
        z1 = 1
        do 31 k = 1,3
        if(r(4,sites(k,i)).gt.1.5) then
        z1 = 0
        y1(1) = r(1,sites(sh(k+1),i))*r(4,sites(sh(k+1),i))
        y1(2) = r(2,sites(sh(k+1),i))*r(4,sites(sh(k+1),i))
        y1(3) = r(3,sites(sh(k+1),i))*r(4,sites(sh(k+1),i))
        y2(1) = r(1,sites(sh(k+2),i))*r(4,sites(sh(k+2),i))
        y2(2) = r(2,sites(sh(k+2),i))*r(4,sites(sh(k+2),i))
        y2(3) = r(3,sites(sh(k+2),i))*r(4,sites(sh(k+2),i))
        end if
   31   continue
        if(z1.eq.1) then
        r_av = (r(4,sites(1,i))+r(4,sites(2,i))+r(4,sites(3,i)))/3.
        mcol = 12.*(r_av - rmin)/(rmax-rmin)+1
        call c3f(legend(1,mcol))
        call bgnpol
        do 32  j = 1,3
        x(1) = r(1,sites(j,i))*r(4,sites(j,i))
        x(2) = r(2,sites(j,i))*r(4,sites(j,i))
        x(3) = r(3,sites(j,i))*r(4,sites(j,i))
        call v3f(x)
   32   continue
        call endpol
        else
        call c3f(bordovec)
        call bgnpol
        do 932  j = 1,3
        r_av=r(4,sites(j,i))
        if(r_av.ge.1.5) then
        x(1)=0
        x(2)=0
        x(3)=0
        else
        x(1) = r(1,sites(j,i))*r_av
        x(2) = r(2,sites(j,i))*r_av
        x(3) = r(3,sites(j,i))*r_av
        endif
        call v3f(x)
  932   continue
        call endpol

        end if

     30 continue
        end if
***************** WIRED MODEL **********
        if(pictmode.eq.1) then
        do 40 i = 1,ntr
        z1 = 1
        do 41 k = 1,3
        if(r(4,sites(k,i)).gt.1.5)then
        z1 = 0
         end if
     41 continue
        if(z1.eq.1)then
        call bgnclo
        do 42  j = 1,3
        mcol = 12.*(r(4,sites(j,i)) - rmin)/(rmax-rmin)+1
        x(1) = r(1,sites(j,i))*r(4,sites(j,i))
        x(2) = r(2,sites(j,i))*r(4,sites(j,i))
        x(3) = r(3,sites(j,i))*r(4,sites(j,i))
        call c3f(legend(1,mcol))
        call v3f(x)
     42 continue
        call endclo
        end if
     40 continue
        end if


***************** SMOOTH MAP MODEL **********
        if(pictmode.eq.3) then
        do 50 i = 1,ntr
        z1 = 1
        do 51 k = 1,3
        if(r(4,sites(k,i)).gt.1.5)then
        z1 = 0
         end if
     51 continue
        if(z1.eq.1)then
        call bgnpol
        do 52  j = 1,3
        mcol = 12.*(r(4,sites(j,i)) - rmin)/(rmax-rmin)+1
        if(mcol.lt.1)mcol = 1
        if(mcol.gt.12)mcol = 12
        x(1) = r(1,sites(j,i))*r(4,sites(j,i))
        x(2) = r(2,sites(j,i))*r(4,sites(j,i))
```

```
      x(3) = r(3,sites(j,1))*r(4,sites(j,1))
      call c3f(legend(1,mcol))
      call v3f(x)
   52 continue
      call endpol
      else
      call c3f(bordovec)
      call bgnpol
      do 952  j = 1,3
      r_av=r(4,sites(j,1))
      if(r_av.ge.1.5) then
      x(1)=0
      x(2)=0
      x(3)=0
      else
      x(1) = r(1,sites(j,1))*r_av
      x(2) = r(2,sites(j,1))*r_av
      x(3) = r(3,sites(j,1))*r_av
       ndif
       all v3f(x)
  952   ntinue
      call endpol

      end if
   50 continue
      end if
      return
      end


      subroutine add_point
     parameter(NORMAL = 1,HOLE = 0)
     parameter(maxst=70000, maxtr=2*maxst)
     parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
      common /points/ nost,r(4,maxst)
       common /net/ ntr,next(3,maxtr),sites(3,maxtr)
     common/mode/ exitmode,pictmode,pointmode
      common/shtri/ sh(4),co(3),con(3)
     common/rots/ angl(7),pov(3)
     common/col/ color,mcolor,legend(3,13)
     common/extremum/rmin,rmax,radius,rad_ex
     integer exitmode,pictmode,pointmode
     integer next,sites,tgl
     logical ok
     nost = nost+1
     r(1,nost) = con(1)
     r(2,nost) = con(2)
     r(3,nost) = con(3)
     if(pointmode .eq. NORMAL) r(4,nost) = radius
     if(pointmode .eq. HOLE) r(4,nost) = 10.0

     tgl =1
     call search(tgl,nost,ok,mk)
     if(ok)then
     call bar(tgl,nost,mk)
     call recon(tgl)
     end if
     return
     end



     subroutine remove_point
     parameter(maxst=70000, maxtr=2*maxst)
     parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
     parameter(piram=.015)
      common /points/ nost,r(4,maxst)
       common /net/ ntr,next(3,maxtr),sites(3,maxtr)
     common/mode/ exitmode,pictmode,pointmode
      common/shtri/ sh(4),co(3),con(3)
     common/rots/ angl(7),pov(3)
     common/col/ color,mcolor,legend(3,13)
     common/extremum/rmin,rmax,radius,rad_ex
     integer exitmode,pictmode,pointmode
     integer next,sites,tgl
     logical ok
      radius = ((color - x1)/w)*(rmax-rmin)+rmin
      r(1,nost+1) = con(1)*radius
      r(2,nost+1) = con(2)*radius
      r(3,nost+1) = con(3)*radius
      r(4,nost+1) = radius
      rad = piram
      tgl = 1

      call search(tgl,nost+1,ok,mk)
      call distance(rad,m,nost+1)
      n = sites(m,tgl)
      do 70 i = n,nost-1
      do 71 k = 1,4
   71   r(k,i) = r(k,i+1)
   70  continue

      nost = nost-1
      tgl = 1
      call tetra

      do 73 np = 5,nost
      call search(tgl,np,ok,mk)
      if(ok)then
      call bar(tgl,np,mk)
      call recon(tgl)
      end if
   73 continue
       mbel = 1
      call setbel(mbel)
      call ringbe
      return
      end


       subroutine distance(rad,kmin,np)
     parameter(maxst=70000, maxtr=2*maxst)
     parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
      common /points/ nost,r(4,maxst)
       common /net/ ntr,next(3,maxtr),sites(3,maxtr)
     common/mode/ exitmode,pictmode,pointmode
      common/shtri/ sh(4),co(3),con(3)
     common/rots/ angl(7),pov(3)
     common/col/ color,mcolor,legend(3,13)
     common/extremum/rmin,rmax,radius,rad_ex
     integer exitmode,pictmode,pointmode
     integer next,sites,tgl
     logical ok
     real x(3),y(3)
      radius = ((color - x1)/w)*(rmax-rmin)+rmin

       dist = 0
```

```
      do 80 k =1,3
      do 81 j = 1,3
      x(j) = r(j,np)*r(4,np)
      y(j) = r(j,sites(k,tgl))*r(4,sites(k,tgl))
      dist1 = dist1+(x(j) - y(j))**2
 81   continue
      if(k.eq.1)then
      dmin = dist1
      kmin = 1
      else if(dist1 .lt.dmin)then
      kmin = k
      dmin = dist1
      end if
 80   continue
      if(dist1 .lt. rad) m = -1
      return
      end
```

```
      subroutine make_change(x4)
    parameter(NORMAL = 1,HOLE = 0)
    parameter(maxst=70000, maxtr=2*maxst)
    parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
    parameter(del = 0.015)
    common/col/ color,mcolor,legend(3,13)
    common/extremum/rmin,rmax,radius,rad_ex
      common/mode/exitmode,pictmode,pointmode
      integer pointmode
      real x4(6)
      if(rad_ex.ge. x4(1).and.rad_ex.lt.x4(2)+del) then
       color = x1
       radius = 10.0
        mcolor = 13
        pointmode = HOLE
        end if
        if(pointmode .eq.NORMAL) then
      if(rad_ex .le.x4(3).and. rad_ex .ge. x4(2)+2*del) then
      rmin = (rad_ex-x1-w/8)*1.5/(0.875*w)
      if(rmin .le .del)rmin = 0
      color = x1
      end if
      if(rad_ex.gt.x4(4)) then
      rmax = (rad_ex-x1-w/8)*1.5/(0.875*w)
      color = x1+w
      end if
      if(rad_ex .lt. x4(4) .and. rad_ex .gt. x4(3)) then
      color = ((rad_ex-x4(3))*w/(x4(4)-x4(3)))+x1
      end if
      end if
      return
      end
```

```
      subroutine make_change_back
    parameter(maxst=70000, maxtr=2*maxst)
    parameter(w = 0.45, h = 0.1,x1 = 1.6,y1 = -0.6,dist = 0.18)
    parameter(NORMAL = 1,HOLE = 0)
    common/col/ color,mcolor,legend(3,13)
      common/mode/exitmode,pictmode,pointmode
    common/extremum/rmin,rmax,radius,rad_ex
      integer pointmode
      pointmode = NORMAL
    radius = ((color - x1)/w)*(rmax-rmin)+rmin
    rad_ex = radius*0.875*w/1.5 + x1+w/8
    return
    end

    function sh(k)
    sh = mod(k+2,3) +1
    return
    end
```

```
      subroutine  int()
*
*     ____________
*     interpolation
*     'nst' - # points, 'r' - their coordinates
*     'ntr' - # triangles
*     radcir - radius**2 of cilcle for each triangle
            parameter(maxst=70000
     .           ,minst= 70000
     .           ,maxtr=2*minst)
       parameter(nbndp=11)
      parameter(nfunc=5)
      common /points/ nst,ntr
      real r(3,maxst)
       integer next(3,maxtr),sites(3,maxtr)
      common/lala/r,next,sites
      real tang(nfunc,minst)
      real radcir(maxtr),wight(minst)
      common/work1/ radcir,wight,tang
      common /ilder/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      parameter(maxlv=20)
      integer work(6*maxlv+9*maxst)
      open(1,file='tempor',status='unknown',form='unformatted')
      open(2,file='addual',status='unknown',form='unformatted')
       open(7,file='triang')
      open(9,file='rescales',status='unknown',form='unformatted')
      open(8,file = 'text3')

      call glider

      read (2,end=1010,err=1010) ntr1,nst1
      read (2,end=1010,err=1010) ((r(i0,i1),i0=1,3),i1=1,nst1)
c     print *,'    ===>',nst1,' points have been read'
 16   format (a,i7,a)
 15   format(a)
      if(nst1.gt.maxst) then
c        print *,'===> dimension maxst is too small <==='
c        print *,'===>    put    maxst = ',nst1,' <==='
         stop 18
         end if
      goto 1020
1010  write(8,*) '40 30'
      write(8,*)'Interpolation is not aloud'
      write(8,*)'The second input parameter is too small'
      close(8)
      call try1("text3")
      return
 1020 continue
      write(8,*) '40  30'
      write(8,*)  ' Interpolation is done '
      write(8,16)' ',nst1,' points have been read'

      read (1,end=1010,err=1010) ntr,nst
      read (1,end=1010,err=1010) ((sites(i4,i5),i4=1,3),i5=1,ntr)
      read (1,end=1010,err=1010) ((next(i2,i3),i2=1,3),i3=1,ntr)
c     print *,'    ===>',ntr,' triangles have been read    <==='



      if(nst.gt.minst) then
         print *,'===> dimension minst is too small <==='
         print *,'===>    put    minst = ',nst ,' <==='
         stop 13
         end if
      if(ipog.ne.0) then
         do 7659 iip=1,nst-nbndp
            read(9,end=1010,err=1010) dum1,dum2,dum3,wight(iip)
7659        continue
         print *,'    ===>  errors have been read  <==='
         endif

      ismt=kva
      ikill=0
      if(ipik.ge.1) ikill=1
      istat=0
      if(ipik.eq.1.and.ismt.eq.1) istat=1
*******************************
*           ismt= 0  - no smoothing
*           ismt= 1  - errors are taken from the file with c r(...)
*           ismt= 2  - errors are equal 1
*           ikill= 0  - don"t kill the peaks
*           ikill= 1  - kill the peaks
*           istat=1   - take into account errors while killing peaks
*           istat=0   - don"t take into account errors while killing p.
*******************************

      if(ipik.lt.0) then
c     print *,' kill the peaks  ?'
c     call getans(ipik)
      if(ipik.ne.0) then
         ikill=1
         if(ipog.ne.0) then
c           print *,' take into account errors ?'
c           call getans(ipog)
            if(ipog.ne.0) istat=1
            endif
         endif
      end if

      if(ismt.lt.0) then
c        print *,' make smoothing  ?'
c        call getans(il)
         if(il.ne.0) then
            ismt=2
            if(ipog.ne.0) then
c              print *,' take into account errors ?'
c              call getans(ipog)
               if(ipog.ne.0) ismt=1
               endif
            endif
         endif
c        print *,ikill ,'       =ikill'

      if(ikill.eq.1) then
      call smstat(istat,r,wight,next,sites,minst,work)
      end if

      if(ismt.ge.1) call smooth(r,wight,next,sites,work,minst,ismt)

      call preint(r,next,sites,radcir,tang,work,minst)

      call interp(ntr1,nst1,r,next,sites,tang,work,maxlv,maxst)

c     print *,'    ===>   write on the disk    <=== '
      call outint(ntr1,nst1,r,work)
      close(7)
      call plider(' interp ')
```

```
      write(8,*)'   Hit right mouse button to continue'
      close(8)
      if (ind.eq.1) then
      call tryl("text3")
      endif
c     print *,'    ===>   end  of  job         <=== '
      close(1)
      close(2)
      close(3)
      close(9)
      return
      end


      subroutine interp(ntrl,nstl,r,next,sites,tang,w,maxlv,maxst)
      integer w(6*maxlv+9*maxst)
      real tang(1)
      real r(1)
      integer next(1),sites(1)
      iio=1
      imo=iio+maxst*2
      iib=imo+maxst*2
      ix=iib+maxst*2
      ior=ix+maxst*2
      ico=ior+maxst
      inn=ico+maxlv*4
      iii=iio+maxst
*     print *,'inn=',inn
*     print *,'total memory =',inn+maxlv*2
      call prepar(ntrl,nstl,r,next,sites,w(ico),w(inn)
     &,w(iio),w(imo),w(iib),w(ix),w(ior),w(iii),maxlv,maxst)

c     print *,'    ===>  interpolation  <===  '
      write(8,15) '     model of surface is constructed'
   15 format(a)
      call addpts(ntrl,nstl,r,sites,tang,w(iio),w(iii))
      return
      end

      subroutine prepar(ntrl,nst,r,next,sites
     &,corner,nn,iord,mord,ibf,x,order,list,maxlv,maxst)
*     ------------------------------
*     ha~alxhaq podgotobka
c'    parameter(maxlv=20)
*     'nst' - # to~ek, 'r' - ix koopdihaty, 'w' - pogpeihoctl.
      common /points/ nst0,ntr0
      real r(3,maxst)
      integer next(3,1),sites(3,1)
      integer  corner(4,maxlv),nn(2,maxlv),iord(maxst,2),mord(maxst,2)
      integer ibf(maxst*2)
      real x(2,maxst),order(maxst)
      integer list(0:1)
c!    equivalence (list(0),iord(1,2))

      ii=int(log(real(maxst))/log(3.0))+1
      if (ii.gt.maxlv)  then
          print *,'===>  change parameters. put maxlv= ',ii,'<==='
          stop
          endif
      nd=nst-nst0
      do 10 j0=1,nd
         x(1,j0)=r(1,j0+nst0)
         x(2,j0)=r(2,j0+nst0)
   10    continue

c     print *,'    ===> points are beeng reordered <===*
      do 40 id=1,2
          do 20 i=1,nd
   20        order(i)=x(id,i)
          call kb07a(order,nd,iord(1,id))
          do 30 mm=1,nd
   30        mord(iord(mm,id),id)=mm
   40     continue


      call uprspl(nd,corner,nn,iord,mord,maxst,maxlv,x,ibf)
      nt=1
      do 50 np=1,nd
          call searchint(nt,iord(np,2)+nst0,r,next,sites)
          order(iord(np,2))=nt
   50     continue
      call kb07a(order,nd,iord)
      nt=ntrl
      list(nt)=nst
      do 60 i3=nst,nst0+1,-1
         ns=i3-nst0
         iord(i3,1)=iord(ns,1)+nst0
 1010    if(order(ns).eq.nt) goto 1020
            nt=nt-1
            list(nt)=i3
            if(nt.ge.0) goto 1010
c           print *,'condition impossible'
            goto 1020
 1020    continue
   60    continue
      do 70 k0=0,nt-1
      list(k0)=nst0
   70 continue
      do 80 i4=1,nst0
   80    iord(i4,1)=i4
      return
      end


      subroutine addpts(ntrl,nstl,r,sites,tang,iord,list)
*     ------------------------------
*
      parameter(nfunc=5)
      integer st,sh
      common /points/ nst,ntr
      real r(3,1)
      integer sites(3,1)
      real tang(nfunc,1)
*     integer iord(maxst),list(0:maxtr)
      integer iord(1),list(0:1)
      real h2(3),a(4,3),c(6,3)
      sh(m)=mod(m,3)+1
      save

      ns=list(0)
      do 50 nt=1,ntr
         nds=list(nt)-ns
         if(nds.gt.0) then
            nsct=nt
            do 40 k0=1,nds
               np=iord(k0+ns)
               call wghts(h2,nt,np,r,sites)

*              ihteppolqciq: bzbe(ehhoe cpedhee ot tpex bep(ih
```

```
*            h2(1)+h2(2)+h2(3)=1

             z=0
             do 30 nv=1,3
                st=sites(nv,nt)
                w=funcal(st,np,r,tang)
                z=z+w*h2(nv)
30           continue
             r(3,np)=z
40           continue
          end if
       ns=list(nt)
50     continue
   return
   end


   subroutine wghts(h,nt,np,r,sites)
*  ---------------------------------
*  h - ploJadi tpex obpazobab[ixcq tpeugolxhikob.
   parameter (eps=1e-8)
   integer st1,st2,st3
   common /points/ nst,ntr
   real r(3,1)
   integer sites(3,1)
   real h(3)
   save

   st1=sites(1,nt)
   st2=sites(2,nt)
   st3=sites(3,nt)
   h1=abs((r(1,st2)-r(1,np))*(r(2,st3)-r(2,np))
  .      -(r(2,st2)-r(2,np))*(r(1,st3)-r(1,np)))
   h2=abs((r(1,st3)-r(1,np))*(r(2,st1)-r(2,np))
  .      -(r(2,st3)-r(2,np))*(r(1,st1)-r(1,np)))
   h3=abs((r(1,st1)-r(1,np))*(r(2,st2)-r(2,np))
  .      -(r(2,st1)-r(2,np))*(r(1,st2)-r(1,np)))
   r1=(r(1,st1)-r(1,np))**2+(r(2,st1)-r(2,np))**2
   r2=(r(1,st2)-r(1,np))**2+(r(2,st2)-r(2,np))**2
   r3=(r(1,st3)-r(1,np))**2+(r(2,st3)-r(2,np))**2
   if(r1.lt.eps) then
      h(1)=1
      h(2)=0
      h(3)=0
   else if(r2.lt.eps) then
      h(1)=0
      h(2)=1
      h(3)=0
   else if(r3.lt.eps) then
      h(1)=0
      h(2)=0
      h(3)=1
   else
      h(1)=(h2**2*r2**2+h3**2*r3**2)*(h1)**2
      h(2)=(h3**2*r3**2+h1**2*r1**2)*(h2)**2
      h(3)=(h1**2*r1**2+h2**2*r2**2)*(h3)**2
      z=h(1)+h(2)+h(3)
      h(1)=h(1)/z
      h(2)=h(2)/z
      h(3)=h(3)/z
      end if
   return
   end


      subroutine searchint(tgl,np,r,next,sites)
*     -----------------------------------------
*     polck tpeugolxhika b kotopyj popadaet to~ka np
      integer tgl,m,np,prev
      common /points/ nst,ntr
      real r(3,1)
      integer next(3,1),sites(3,1)
      save

      mk=0
      a3=volum(r(1,np),r(1,sites(1,tgl)),r(1,sites(2,tgl)))
      a1=volum(r(1,np),r(1,sites(2,tgl)),r(1,sites(3,tgl)))
      a2=volum(r(1,np),r(1,sites(3,tgl)),r(1,sites(1,tgl)))
      if(a1.ge.0.and.a2.ge.0.and.a3.ge.0) then
          return
          end if
      if(a2.lt.a3) then
          if(a1.lt.a2) then
              m=1
          else
              m=2
              end if
      else
          if(a1.lt.a3) then
              m=1
          else
              m=3
              end if
          end if
      k0=0
1010  prev=tgl
      tgl=next(m,tgl)
      m2=markint(tgl,prev,next)
      m3=mod(m2,3)+1
      m1=mod(m3,3)+1
      a3=volum(r(1,np),r(1,sites(m1,tgl)),r(1,sites(m2,tgl)))
      a1=volum(r(1,np),r(1,sites(m2,tgl)),r(1,sites(m3,tgl)))
      if(a1.lt.a3) then
          m=m1
          a=a1
      else
          m=m3
          a=a3
          end if
      k0=k0+1
      if(k0.gt.ntr) then
          print *,' ===> triangle not found  <===='
          print *,' ===>   fatal error       <===='
          stop 31
          end if
      if(a.lt.0) goto 1010
      return
      end


      subroutine outint(ntr1,nst1,r,sites)
*     ------------------------------------
*     bybod dlq bizuallizacii
*!        sites -work arr. dimension = maxst*6
      common /points/ nst,ntr
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &               ,ipik,ipog
      real r(3,1)
```

```
      logical frame
      integer sites(3,1)
      save
      r1=0.5
c     write (7,*) ntr1,nst1
      do 220 j1=1,nst1
        if(r(3,j1).lt.0.0.and.r(3,j1).gt.-9 ) r(3,j1)=0
        if(r(3,j1).gt.zmax ) r(3,j1)= zmax
 220  continue
      read(2)((sites(i3,i4),i3=1,3),i4=1,ntr1)
        nt=0
      do 712 it=1,ntr1
      if(frame(sites(1,it),r)) goto 712
      if(frame(sites(2,it),r)) goto 712
      if(frame(sites(3,it),r)) goto 712
        nt=nt+1
 712  continue
      write (7,*) nt,nst1
      write(8,16) '          ' ,nt,' triangles'
      write(8,16) '          ',nst1,' points'
      write(8,15) '            job is done'
 16   format (a,i7,a)
 15   format(a)
      do 10 j1=1,nst1
        if(r(1,j1).ge.0.0 .and.r(1,j1).le.1.0) then
        if(r(2,j1).ge.0.0 .and.r(2,j1).le.1.0) then
        write(7,*)r(1,j1),r(2,j1),r(3,j1)
        else
        write(7,*)r1,r1,r(3,j1)
        end if
        else
        write(7,*)r1,r1,r(3,j1)
        end if
  10      continue
      na=0
      do 710 it=1,ntr1
      if(frame(sites(1,it),r)) goto 710
      if(frame(sites(2,it),r)) goto 710
      if(frame(sites(3,it),r)) goto 710
      write (7,*) sites(1,it),sites(2,it),sites(3,it)
 710  continue
      return
      end

*_____________________________________________________________
      function funcal(ns,ms,r,tang)
      parameter(nfunc=5)
      real tang(nfunc,1),phi(nfunc),r(3,1)
      call funcs(r(1,ms)-r(1,ns),r(2,ms)-r(2,ns),phi)
      funcal=r(3,ns)
      do 1 k=1,nfunc
1     funcal=funcal+dprod(tang(k,ns),phi(k))
      return
      end
*_____________________________________________________________
      subroutine calm(rmat,rhs,r2)
      parameter(nfunc=5)
      real phi(nfunc),rmat(nfunc,nfunc),rhs(nfunc),r2(3)
      sum=1.d0/(dprod(r2(1),r2(1))+
     *      dprod(r2(2),r2(2))+dprod(r2(3),r2(3)))
      sum=sum**2
      call funcs(r2(1),r2(2),phi)
      do 1 k=1,nfunc
      rhs(k)=rhs(k)+(sum*dprod(r2(3),phi(k)))
      do 2 l=1,nfunc
      rmat(k,l)=rmat(k,l)+(sum*dprod(phi(k),phi(l)))
2     continue
1     continue
      return
      end
*_____________________________________________________________
      subroutine clrm(rmat,rhs)
      parameter(nfunc=5)
      real rmat(nfunc,nfunc),rhs(nfunc)
      do 1 k=1,nfunc
      rhs(k)=0d0
      do 1 l=1,nfunc
      rmat(k,l)=0d0
1     continue
      return
      end
*_____________________________________________________________
      subroutine funcs(x,y,phi)
      parameter(nfunc=5)
      real phi(nfunc)
      phi(1)=x
      phi(2)=y
      phi(3)=dprod(x,x)
      phi(4)=dprod(x,y)
      phi(5)=dprod(y,y)
      return
      end
*_____________________________________________________________
*_____________________________________________________________
*_____________________________________________________________

* common subprograms for interpolation
      subroutine preint(r,next,sites,radcir,tang,work,minst)
      real r(1),next(1),sites(1),radcir(1),tang(1),work(1)
      call calrad(r,sites,radcir)
      call calnor(r,next,sites,radcir,tang,work,minst)
      return
      end


      subroutine calnor(r,next,sites,radcir,tang,mask,minst)
*     _____________
*     papametpy okpuvhocti, opicahhoj bokpug modeli
*     rbnd - padiuc,  nbnbp - kol-bo to~ek ha okpuvhocti
      parameter(nbndp=11,rbnd=10.)
      parameter(nfunc=5)
      integer sh,st
      common /points/ nst,ntr
      real r(3,1)
      integer next(3,1),sites(3,1)
*     radcir - kbadpat padiuca opicahhoj bokpug tp-ka okpuvhocti
      real radcir(1)
      real tang(nfunc,1)
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &               ,ipik,ipog
      integer mask(1),crv
* mask - work arr. dim = minst
      real r2(3),rnor(nfunc,nfunc)
      logical frame
      sh(m)=mod(m,3)+1
      save
```

```
c     print *,'nst,ntr ', nst,ntr
      do 10 ns=1,nst
         if(ns.lt.nst-nbndp) then
            mask(ns)=0
         else
            mask(ns)=1
            end if
 10            continue
      do 30 nt=1,ntr
         do 20 i0=1,3
         st=sites(i0,nt)
         if(mask(st).eq.0) then
*           obojti bokpug bepiihy  sites(i0,nt)
            mask(st)=1
            neb=next(sh(i0),nt)
            lst=nt
            crv=0
            call clrm(rnor,tang(1,st))
 1010        continue
                if(crv.gt.minst/8) then
                print *,'    ===>   fatal error         <==='
                stop 21
                end if
              m3=markint(neb,lst,next)
              m1=sh(m3)
              m2=sh(m1)
*             by=iciqetcq hopmaix k tpeugoixhiku
              lst=neb
              neb=next(m2,neb)
* first neighbor
c             if(radcir(lst).lt.rdmax0**2) then
*let us try this
c             if(.not.frame(sites(m3,lst),r)) then
              if(.true.)then
              r2(1)=r(1,sites(m3,lst))-r(1,st)
              r2(2)=r(2,sites(m3,lst))-r(2,st)
              r2(3)=r(3,sites(m3,lst))-r(3,st)
                 crv=crv+1
                 call calm(rnor,tang(1,st),r2)
                 end if
* second neighbor
              lst1=next(m1,lst)
c             if(lst1.gt.0.and.radcir(lst1).lt.rdmax0**2) then
              if(lst1.gt.0) then
              m3=markint(lst1,lst,next)
*let us try this
c             if(.not.frame(sites(m3,lst1),r)) then
              if(.true.)then
              r2(1)=r(1,sites(m3,lst1))-r(1,st)
              r2(2)=r(2,sites(m3,lst1))-r(2,st)
              r2(3)=r(3,sites(m3,lst1))-r(3,st)
                 crv=crv+1
                 call calm(rnor,tang(1,st),r2)
                 end if
               end if
            if(lst.ne.nt) goto 1010
            if(crv.le.5) then
             if(crv.le.1) then
                call clrm(rnor,tang(1,st))
                print *,crv,st,nt
                else
                 call  gauss}(rnor,2,nfunc,tang(1,st),1,1)
                 do 86 k1=3,nfunc
86               tang(k1,st)=0d0
                    endif
                else
                call  gauss}(rnor,nfunc,nfunc,tang(1,st),1,1)
                end if
                end if
 20          continue
 30       continue
      return
      end

      subroutine smooth(r,wight,next,sites,w,minst,ismt)
*     ______
      integer w(minst*3)
      real r(3,1),wight(1)
      integer next(3,1),sites(3,1)
*     dimension temp(minst),nmem(minst/2),rmem(minst/2)
*     integer mask(1)
      itemp=1
      inmem=itemp+minst
      irmem=inmem+minst/2
      imask=irmem+minst/2
      call smootl(r,wight,next,sites,w(itemp)
     & ,w(inmem),w(irmem),w(imask),minst,ismt)
      return
      end

      subroutine smootl(r,w,next,sites,temp,nmem,rmem,mask,minst,ismt)
*     ______
      parameter (eps=1e-8)
*     papametpy okpuvhocti, opicahhoj bokpug modeli
*     rbnd - padiuc,  nbnbp - kol-bo to-ek ha okpuvhocti
      parameter (nbndp=11,rbnd=10.)
      integer sh,st,crv
      common /points/ nst,ntr
      real r(3,1),w(1)
      integer next(3,1),sites(3,1)
      dimension temp(minst),nmem(minst/2),rmem(minst/2)
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &               ,ipik,ipog
      integer mask(1)
      real b1,b2
      sh(m)=mod(m,3)+1
      save
      if(ismt.ne.1.and.ismt.ne.2) return
c     print *,'    ===>        smoothing       <==='
      if(ismt.eq.2) w(1)=1
      do 10 ns=1,nst
         if(ns.lt.nst-nbndp) then
            mask(ns)=0
         else
            mask(ns)=1
            end if
         temp(ns)=r(3,ns)
         nmem(ns)=0
 10      continue
      do 40 nt=1,ntr
         do 30 i0=1,3
            st=sites(i0,nt)
            if(mask(st).eq.0) then
*              obojti bokpug bepiihy  sites(i0,nt)
               mask(st)=1
               if(ismt.eq.2.or.w(st).gt.eps) then
                  neb=next(sh(i0),nt)
```

```
                lst=nt
                crv=0
 1010           if(crv.gt.minst/8) then
                   print *,'   ===>    fatal error      <===='
                   stop 21
                   end if
                crv=crv+1
                m3=markint(neb,lst,next)
                m1=sh(m3)
                m2=sh(m1)
                lst=neb
                neb=next(m2,neb)
                nmem(crv)=sites(m3,lst)
                rmem(crv)=(r(1,st)-r(1,nmem(crv)))**2
                        +(r(2,st)-r(2,nmem(crv)))**2
                if(lst.ne.nt) goto 1010
                b1=0
                b2=0
                do 20 ic=1,crv
                   r2=(r(1,st)-r(1,nmem(ic)))**2
                     +(r(2,st)-r(2,nmem(ic)))**2
                   if(r2.lt.3*rdmax0**2) then
                      rcorr=rdmax0**2*0.1
                      if(r2.gt.rcorr/16) then
c                        rd=2*sqrt(r2/rmed)
                         rd=sqrt(r2/rcorr)-0.25
                      else
                         rd=0
                         end if
                      if(ismt.eq.1) then
                         den=1/(w(st)+(w(st)+w(nmem(ic)))*rd)
                      else
                         den=1/(w(1)+(w(1)+w(1))*rd)
                         endif
                      b1=b1+r(3,nmem(ic))*den
                      b2=b2+den
                      end if
   20              continue
                if(ismt.eq.1) then
                   temp(st)=(r(3,st)+w(st)*b1)/(1+w(st)*b2)
                else
                   temp(st)=(r(3,st)+w(1)*b1)/(1+w(1)*b2)
                   endif
             else
                temp(st)=r(3,st)
                end if
             end if
   30     continue
   40   continue
        do 50 n0=1,nst-nbndp
c          if(abs(temp(n0)-r(3,n0)).le.3*w(n0)) then
              r(3,n0)=temp(n0)
c             end if
   50   continue
        return
        end




        integer function markint(tr1,tr2,next)
*       ---------------------------------------
        integer tr1,tr2
        integer next(3,1)

*       oppedelehie opiehtacii b tp-ke tr1 po tp-ku tr2
        do 10 markint=1,3
           if(next(markint,tr1).eq.tr2) return
   10      continue
        print *,'     ===>  "next" is in disaccord       <===='
        print *,'     ===>       fatal error             <===='
        stop 45
        end



        subroutine cairad(r,sites,radcir)
*       ---------------------------------
*       by-iciq'tcq padiycy opicahhyx bokpyg tpeygolxhikob okpyvhocte)
        integer tg1,st1,st2,st3
        common /points/ nst,ntr
        real r(3,1)
        integer sites(3,1)
        real radcir(1)
        real*8 a(2,2),b(2),det,pp
        real center(2)
        save
        pp(x)=dprod(x,x)

        do 10 tg1=1,ntr
        st1=sites(1,tg1)
        st2=sites(2,tg1)
        st3=sites(3,tg1)
        a(1,1)=r(1,st2)-r(1,st1)
        a(1,2)=r(2,st2)-r(2,st1)
        a(2,1)=r(1,st3)-r(1,st1)
        a(2,2)=r(2,st3)-r(2,st1)
        b(1)=(pp(r(1,st2))+pp(r(2,st2))-pp(r(1,st1))-pp(r(2,st1)))/2
        b(2)=(pp(r(1,st3))+pp(r(2,st3))-pp(r(1,st1))-pp(r(2,st1)))/2
        det=a(1,1)*a(2,2)-a(2,1)*a(1,2)
        center(1)=(a(2,2)*b(1)-a(1,2)*b(2))/det
        center(2)=(a(1,1)*b(2)-a(2,1)*b(1))/det
        radcir(tg1)=pp(r(1,st1)-center(1))+pp(r(2,st1)-center(2))
   10   continue
        return
        end

        subroutine smstat(istat,r,wight,next,sites,minst,w)
        integer w(3*minst)
        real r(3,1),wight(1)
        integer next(3,1),sites(3,1)
        itek=1
        id=itek+minst/8
        ismot=id+minst/8
        ic=ismot+minst
        inei=ic+minst/4+1
        imask=inei+minst/8

        call smstal(istat,r,wight,next,sites,w(itek),w(id   ),w(ismot),
     .            w(ic),w(inei ),w(imask),minst)

        return
        end

        subroutine smstal(istat,r w,next,sites,tek,d,smot,c,nei,mask,
     .            minst)
*       ------------------
```

```
                lst=nt
                crv=0
 1010           if(crv.gt.minst/8) then
                   print *,'    --->    fatal error      <---'
                   stop 21
                   end if
                crv=crv+1
                m3=markint(neb,lst,next)
                m1=sh(m3)
                m2=sh(m1)
                lst=neb
                neb=next(m2,neb)
                nmem(crv)=sites(m3,lst)
                rmem(crv)=(r(1,st)-r(1,nmem(crv)))**2
                        +(r(2,st)-r(2,nmem(crv)))**2
                if(lst.ne.nt) goto 1010
                b1=0
                b2=0
                do 20 ic=1,crv
                   r2=(r(1,st)-r(1,nmem(ic)))**2
                     +(r(2,st)-r(2,nmem(ic)))**2
                   if(r2.lt.3*rdmax0**2) then
                      rcorr=rdmax0**2*0.1
                      if(r2.gt.rcorr/16) then
c                        rd=2*sqrt(r2/rmed)
                         rd=sqrt(r2/rcorr)-0.25
                      else
                         rd=0
                         end if
                      if(ismt.eq.1) then
                         den=1/(w(st)+(w(st)+w(nmem(ic)))*rd)
                      else
                         den=1/(w(1)+(w(1)+w(1))*rd)
                         endif
                      b1=b1+r(3,nmem(ic))*den
                      b2=b2+den
                      end if
   20              continue
                if(ismt.eq.1) then
                   temp(st)=(r(3,st)+w(st)*b1)/(1+w(st)*b2)
                else
                   temp(st)=(r(3,st)+w(1)*b1)/(1+w(1)*b2)
                   endif
             else
                temp(st)=r(3,st)
                end if
             end if
   30        continue
   40     continue
          do 50 n0=1,nst-nbndp
c            if(abs(temp(n0)-r(3,n0)).le.3*w(n0)) then
               r(3,n0)=temp(n0)
c            end if
   50     continue
          return
          end



          integer function markint(tr1,tr2,next)
          integer tr1,tr2
          integer next(3,1)

          oppedelehie opiehtacii b tp-ke tr1 po tp-ku tr2
          do 10 markint=1,3
             if(next(markint,tr1).eq.tr2) return
   10        continue
          print *,'    --->  "next" is in disaccord       <---'
          print *,'    --->       fatal error             <---'
          stop 45
          end



          subroutine calrad(r,sites,radcir)
          by-iclq'tcq padlycy opicahhyx bokpyg tpeygolxhikob okpyvhocte}
          integer tg1,st1,st2,st3
          common /points/ nst,ntr
          real r(3,1)
          integer sites(3,1)
          real radcir(1)
          real*8 a(2,2),b(2),det,pp
          real center(2)
          save
          pp(x)=dprod(x,x)

          do 10 tg1=1,ntr
          st1=sites(1,tg1)
          st2=sites(2,tg1)
          st3=sites(3,tg1)
          a(1,1)=r(1,st2)-r(1,st1)
          a(1,2)=r(2,st2)-r(2,st1)
          a(2,1)=r(1,st3)-r(1,st1)
          a(2,2)=r(2,st3)-r(2,st1)
          b(1)=(pp(r(1,st2))+pp(r(2,st2))-pp(r(1,st1))-pp(r(2,st1)))/2
          b(2)=(pp(r(1,st3))+pp(r(2,st3))-pp(r(1,st1))-pp(r(2,st1)))/2
          det=a(1,1)*a(2,2)-a(2,1)*a(1,2)
          center(1)=(a(2,2)*b(1)-a(1,2)*b(2))/det
          center(2)=(a(1,1)*b(2)-a(2,1)*b(1))/det
          radcir(tg1)=pp(r(1,st1)-center(1))+pp(r(2,st1)-center(2))
   10     continue
          return
          end

          subroutine smstat(istat,r,wight,next,sites,minst,w)
          integer w(3*minst)
          real r(3,1),wight(1)
          integer next(3,1),sites(3,1)
          itek=1
          id=itek+minst/8
          ismot=id+minst/8
          ic=ismot+minst
          inei=ic+minst/4+1
          imask=inei+minst/8

          call smstal(istat,r,wight,next,sites,w(itek),w(id   ),w(ismot),
                      w(ic),w(inei ),w(imask),minst)

          return
          end

          subroutine smstal(istat,r,w,next,sites,tek,d,smot,c,nei,mask,
                      minst)
```

```
*     smstal cha4ala fopm!puet maccib tek, pepbye  ntek !lemehtob
*     k-pogo codepvat zha-ehiq z-koopdihat b camoj to~ke i ee co-
*     cedqx l popqdka.
      parameter(nbndp=11,rbnd=10.)
      integer sh,crv,st
      real r(3,1),w(1)
      integer next(3,1),sites(3,1)
      common /points/nst,ntr
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg.miner,kva
     &               ,ipik,ipog
*
      dimension tek(minst/8),d(minst/8),
     .                c(minst/4+1),nei(minst/8)
      integer mask(1)
      real smot(1)
*
       sh(m)=mod(m,3)+1
      save
      do 10 ns=1,nst
         if(ns.lt.nst-nbndp) then
            mask(ns)=0
         else
            mask(ns)=1
            end if
   10    continue
      do 80 nt=1,ntr
        do 801 ii=1,2
        do 801 jj=1,3
        rr=r(ii,(sites(jj),nt)))
        if(rr.lt.(0.).or.rr.gt.(1.)) go to 80
801   continue
          do 70 i0=1,3
             st=sites(i0,nt)
             if(mask(st).eq.0) then
*               obojti bokpug bepi!hy  sites(i0,nt)
*
                neb=next(sh(i0),nt)
                lst=nt
                crv=0
1010            crv=crv+1
                   if(neb.eq.0) goto 80
                if(crv.gt.minst/8) then
                   print *,'   ===>    fatal error      <===='
                   stop 21
                   end if
                m3=markint(neb,lst,next)
                m1=sh(m3)
                m2=sh(m1)
*               zapomiha'tcq homepa cocedej
                nei(  crv)=sites(m3,neb)
*     he o~ehx p'oxoe  mecto
*               if(next(m1,neb).lt.1) goto 80
*               nei(  crv)=sites(markint(next(m1,neb),neb),next(m1,neb))

                lst=neb
                neb=next(m2,neb)
                if(lst.ne.nt) goto 1010

* cocedej coc-itali.   poi!i c~itatx maccib zha~ehij b coc. to~kax

      mask(st)=1
      ntek=0
      do 117 inst=1,crv
      ras=(r(1,st)-r(1,nei(  inst)))**2+(r(2,st)-r(2,nei(  inst)))**2
      if(ras.lt.3*rdmax0) then
         ntek=ntek+1
*           rcorr=rdmax0**2/160
*            if(ras.lt.rcorr) then
      tek(ntek)=r(3,nei(  inst))
      d(ntek)=w(nei(  inst))
*           else
*           tek(ntek)=r(3,st)+(r(3,nei(  inst))-r(3,st))*sqrt(rcorr/ras)
*           d  (ntek)=w(nei(  inst))
*            end if
        end if
117   continue
      ntek=ntek+1
      tek(ntek)=r(3,st)
      d(ntek)=w(st)

*   maccib tek cfopmipobah

      call huber(istat,d,ntek  ,tek,ntek,sko,cent,c,2*ntek+1)
*           _____ huber delaet ocehku pazbpoca zha~ehij z b zohe bli-
*           qhiq to~ki i ocehku camogo zha~ehiq b to~ke,udalqq
*           bybpoc b to~ke ( ppi heobxodimocti )

      smot(st)=cent
      end if
   70         continue
   80     continue
      xvyb=0.
      do 99 st=1,minst
      if(mask(st).ne.0) then
      if(istat.eq.1) sko=w(st)
      if(abs(smot(st)-r(3,st)).ge.(3.*sko)) kvyb=kvyb+1
      if(abs(smot(st)-r(3,st)).ge.(3.*sko)) r(3,st)=smot(st)
      end if
99    continue
*     print *,' ',kvyb,' podablehhyx pikob i/ili kbazipikob  '
      print *,'   ===>    peaks are killed    <===='
      return
      end
      subroutine huber(istat,d,l,a,n,sko,center,c,mc)
*             huber delaet ocehku pazbpoca zha~ehij bybopki a(n)
*             i ocehku cpedhego zha~ehiq bybopki,udalqq
*             bybpoc ( ppi heobxodimocti )

      real c(mc),d(1),a(n),sko,center,h

      if(n.eq.1) then
         center=a(n)
         sko=d(n)
         return
        end if
*
*     upopqdo~im bybopku a(1).........a(n)
      call sort(a,n)
*     tepepx imeem: a(1)<=a(2)<=..........a(n-1)<=a(n)

      if(istat.eq.0) then
      m=n/2
      if((m*2).eq.n) then
            center=(a(m)+a(m+1))*0.5
                   else
            center=a(m+1)
      end if
```

```
*  itak : center - mediaha bybopki a(n)

      do 1 i=1,n
1     c(i)=(a(i)-center)**2
*  c(n) - habop kbadpatob otklohehij ilemehtob a(i) ot mediahy
      call sort(c,n)
      sko=0.
      if(n.ge.4) then
*  ecli ob"em bybopki n >= 4, to otbpacybaem n/4 kpajhix zha-
*  -ehij cbepxu i ctolxko ve chizu:

        n4=n/4
        do 112 i=n4+1,n-n4-1
 112    sko=sko+c(i)
        sko=sko/(n-n4-n4)
                 else
             if(n.eq.3) then
*  ecli ob"em bybopki n = 3, to otbpacybaem odho kpajhee zha-
*  -ehie cbepxu i odho chizu:
               sko=c(2)
              else
*  ecli ob"em bybopki n = 2, to bepem polucummu:
                sko=0.5*(c(1)+c(2))
             end if
      end if
      else
      call sort(d,l)
      sko=0.
      if(l.ge.4) then
*  ecli ob"em bybopki l >= 4, to otbpacybaem l/4 kpajhix zha-
*  -ehij cbepxu i ctolxko ve chizu:

        l4=l/4
        do 212 i=l4+1,l-l4-1
 212    sko=sko+d(i)
        sko=sko/(l-l4-l4)
                 else
             if(l.eq.3) then
*  ecli ob"em bybopki l = 3, to otbpacybaem odho kpajhee zha-
*  -ehie cbepxu i odho chizu:
               sko=d(2)
              else
*  ec.i ob"em bybopki l = 2, to bepem polucummu:
                sko=0.5*(d(1)+d(2))
             end if
      end if
      end if
      sko=sqrt(sko)

* po.agaem papametp h pabhym tpem by-iclehhym cp.-kbadp. otklohehiqm:
      h=sko*3.
*
* ta.ee poohoditcq minimizaciq funkcii xx'bepa:
*
*  center= argmin {  ro(t,a,.....) }
*          minimym bepetcq po ckal. beli-ihe t.
*          ro - f:hkciq xx'bepa,by-iclqetcq b function ro.

      n=0
      do 2 i=1,n
      n=n+1
      c(n)=a(i)-h
      n=n+1
      c(n)=a(i)+h
2     continue
      call sort(c,m)
      i1=1
      i2=m
      del=de(c(1),a,n,h)
      de2=de(c(m),a,n,h)
8     if(i2-i1.le.1) go to 9
      is=(i1+i2)/2
      des=de(c(is),a,n,h)
      if(des) 4,10,5
10    center=c(is)
      return
4     i1=is
      del=des
      go to 8
5     i2=is
      de2=des
      go to 8
9     continue
      center=c(i1)-(c(i2)-c(i1))*del/(de2-del)
* itak, ickomaq bybpocouctoj-ibaq ocehka cpedhego zha-ehiq
* bybopki a(1)....a(n) by-icleha i pabha center
      return
      end
      real function de(t,a,n,h)
      real a(n),h,t,ta
      de=0.
      do 1 i=1,n
      ta=t-a(i)
      if(abs(ta).gt.h) then
         if(ta.gt.h) then
            de=de+h
                  else
            de=de-h
         end if
                    else
         de=de+ta
      end if
1     continue
      return
      end

      subroutine sort(a,n)
*     ____________________
*     podppogpamma sort delaet coptipobku maccika  a(n) po bo.pactahi'
      real a(n)
      if(n.eq.1) return
      do 1 i=1,n-1
      do 1 j=i+1,n
      if(a(i)-a(j)) 1,1,11
11    t=a(i)
      a(i)=a(j)
      a(j)=t
1     continue
      return
      end
*_______________________________________________________________________
      SUBROUTINE GAUSSJ(A,N,NP,B,M,MP)
        implicit real*8(a-h,o-z)
      PARAMETER (NMAX=50)
      real A(NP,NP),B(NP,MP)
      integer IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
      DO 11 J=1,N
```

```
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (ABS(A(J,K)).GE.BIG)THEN
                  BIG=ABS(A(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                do 48 kk=1,N
48              b(kk,1)=0
                return
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            DUM=A(IROW,L)
            A(IROW,L)=A(ICOL,L)
            A(ICOL,L)=DUM
14        CONTINUE
          DO 15 L=1,M
            DUM=B(IROW,L)
            B(IROW,L)=B(ICOL,L)
            B(ICOL,L)=DUM
15        CONTINUE
        ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (A(ICOL,ICOL).EQ.0.)  then
                do 88 kk=1,N
88              b(kk,1)=0
                return
                endif
        PIVINV=1./A(ICOL,ICOL)
        A(ICOL,ICOL)=1.
        DO 16 L=1,N
          A(ICOL,L)=A(ICOL,L)*PIVINV
16      CONTINUE
        DO 17 L=1,M
          B(ICOL,L)=B(ICOL,L)*PIVINV
17      CONTINUE
        DO 21 LL=1,N
          IF(LL.NE.ICOL)THEN
            DUM=A(LL,ICOL)
            A(LL,ICOL)=0.
            DO 18 L=1,N
              A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18          CONTINUE
            DO 19 L=1,M
              B(LL,L)=B(LL,L)-B(ICOL,L)*DUM
19          CONTINUE
          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
          DO 23 K=1,N
            DUM=A(K,INDXR(L))
            A(K,INDXR(L))=A(K,INDXC(L))
            A(K,INDXC(L))=DUM
23        CONTINUE
        ENDIF
24    CONTINUE
      RETURN
      END
```

```
      subroutine ints(ind5,ind6)
*     ____________
*     ihteppolqciq i by~iclehie koopdihat hobyx to~ek
            parameter(minst=70000,maxst=minst, maxtr=2*maxst)
      integer next,sites,tgl
*     'nst' - # to~ek, 'r' - ix koopdihaty
      common /points/ nst,r(4,maxst)
*     'ntr' - # tpeugolxhikob, '/net/' - cpicok cmevhocti diq
*     gpafa dualxhogo k tpiahqulqcli
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)


      if(ind5 .eq. 1) then
      open(8,file = 'text')
      write(8,*) '50 50'
      write(8,*) 'Interpolation is not aloud'
      write(8,*) 'Second input parameter is too small'
      close(8)
      call tryl("text")
      return
      end if

         open(1,file='stempor',form='unformatted',status='unknown')
         open(2,file='saddual',form='unformatted',status='unknown')
      open(7,file='triang')

      read (2,end=1010,err=1010) ntr1,nst1
      read (2,end=1010,err=1010) ((r(i0,i1),i0=1,4),i1=1,nst1)
ccc     print *,'   --->',nst1,' points have been read    <---'
      if(nst1.gt.maxst) then
          print *,'   ---> maxst too small, change and recompile  <---'
          stop 11
          end if
      goto 1020
 1010 print *,'    ---> error reading  <---  '
      stop 12
 1020 continue

      read (1,end=1010,err=1010) ntr,nst0
      read (1,end=1030,err=1030) ((sites(i4,i5),i4=1,3),i5=1,ntr)
      read (1,end=1030,err=1030) ((next(i2,i3),i2=1,3),i3=1,ntr)
ccc     print *,'   --->',ntr,' triangles have been read    <---'
      if(ntr.gt.maxtr) then
          print *,'   ---> maxtr too small, change and recompile  <---'
          stop 13
          end if
      goto 1040
 1030 print *,'    ---> error reading  <---  '
      stop 14
 1040 continue
      if(nst0.gt.minst) then
          print *,'   ---> minst too small, change and recompile  <---'
          stop 15
          end if
      if(nst0.ge.nst1) then
          print *,'   ---> there are no added points  <---'
          stop 16
          end if

      nst=nst0
*     by~iclehie papametpob ihteppolqciohhyx fuhkcij
      call coeff
*     sopipobka to~ek po tpeugolxhikam
ccc     print *,'    --->   sorting   <---  '
      call prepar(ntr,nst1)
ccc     print *,'    --->  interpolation  <---  '
*     by~iclehie koopdihat hobyx to~ek
*     nst1 - hoboe polhoe k~bo to~ek pobepxhocti
*     nst0 - ctapoe k~bo to~ek pobepxhocti
      call addpts(ntr1,nst1)

ccc     print *,'    --->    ok' i write   <--- '
      if(ind6.eq.0)then
      call outint(ntr1,nst1)
      end if
      if (ind6.eq.1)then
      call outint1(ntr1,nst1)
      end if
      close(1)
      close(2)
      return
      end

      subroutine coeff
*     ____________
*     by~iclehie pepepob b bep(ihax opopho) pe(etki
          parameter(minst=70000,maxst=minst, maxtr=2*maxst)
      integer next,sites,sh,crv,st
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      real rnor(3),tang(3),sig(3),r1(3),r2(3)
      common /repers/ e(3,3,minst),derv(2:3,minst)
      common mask(minst)
      sh(m)=mod(m,3)+1
      save

      do 10 ns=1,nst
          mask(ns)=0
   10     continue
      do 100 nt=1,ntr
          do 90 i0=1,3
              st=sites(i0,nt)
              if(mask(st).eq.0) then
*                 obojti bokpuq bep(ihy  sites(i0,nt)
*                 rnorm - hopmalx k kacatelxhoj plockocti
                  do 20 k0=1,3
                     rnor(k0)=0
 20                  continue
                  neb=next(sh(i0),nt)
                  lst=nt
                  crv=0
 1010             crv=crv+1
                  if(crv.gt.minst/4) then
                      print *,'   --->    fatalxhaq   o(ibka      <---'
                      stop 21
                      end if
                  m3=mark(neb,lst)
                  m1=sh(m3)
                  m2=sh(m1)
*                 by~iclqetcq hopmalx k tpeugolxhiku
                  if(crv.eq.1) then
                     r1(1)=r(1,sites(m2,neb))-r(1,st)
                     r1(2)=r(2,sites(m2,neb))-r(2,st)
                     r1(3)=r(3,sites(m2,neb))-r(3,st)
                     absr1=sqrt(r1(1)**2+r1(2)**2+r1(3)**2)
                     r1(1)=r1(1)/absr1
                     r1(2)=r1(2)/absr1
                     r1(3)=r1(3)/absr1
```

```
                else
                   r1(1)=r2(1)
                   r1(2)=r2(2)
                   r1(3)=r2(3)
                   end if
                r2(1)=r(1,sites(m3,neb))-r(1,st)
                r2(2)=r(2,sites(m3,neb))-r(2,st)
                r2(3)=r(3,sites(m3,neb))-r(3,st)
                absr2=sqrt(r2(1)**2+r2(2)**2+r2(3)**2)
                r2(1)=r2(1)/absr2
                r2(2)=r2(2)/absr2
                r2(3)=r2(3)/absr2
                rnor(1)=rnor(1)+
     .             (r1(2)*r2(3)-r1(3)*r2(2)-rnor(1))/crv
                rnor(2)=rnor(2)+
     .             (r1(3)*r2(1)-r1(1)*r2(3)-rnor(2))/crv
                rnor(3)=rnor(3)+
     .             (r1(1)*r2(2)-r1(2)*r2(1)-rnor(3))/crv
                lst=neb
                neb=next(m2,neb)
                if(lst.ne.nt) goto 1010
*               e - pepep b bep[iha
                jmin=0
                cmin=1
                do 30 j1=1,3
                   e(1,j1,st)=r(j1,st)
                   if(abs(e(1,j1,st)).lt.cmin) then
                      jmin=j1
                      cmin=abs(e(1,j1,st))
                      end if
                   sig(j1)=0
   30              continue
                sig(jmin)=1
                do 40 j2=1,3
                   tang(j2)=e(1,sh(j2),st)*sig(sh(j2+1))
     .                     -e(1,sh(j2+1),st)*sig(sh(j2))
   40              continue
                gl=sqrt(scal(tang,tang))
                do 50 j3=1,3
                   e(3,j3,st)=tang(j3)/gl
   50              continue
                do 60 j4=1,3
                   e(2,j4,st)=e(3,sh(j4),st)*e(1,sh(j4+1),st)
     .                       -e(3,sh(j4+1),st)*e(1,sh(j4),st)
   60              continue
                do 80 j6=1,3
                   tang(j6)=0
                   do 70 j5=1,3
                      tang(j6)=tang(j6)+e(j6,j5,st)*rnor(j5)
   70                 continue
   80              continue
                if(tang(1).gt.1e-10) then
                   derv(2,st)=-tang(2)/tang(1)*r(4,st)
                   derv(3,st)=-tang(3)/tang(1)*r(4,st)
                else
                   derv(2,st)=-tang(2)*1e10
                   derv(3,st)=-tang(3)*1e10
                   print *,' ===> very large derivatives <==='
                   end if
                end if
   90         continue
  100      continue
           return
           end

           subroutine prepar(ntr,nst)
*
*          ------------------------------
*          ha-alxhaq podgotobka
                parameter(minst=70000,maxst=minst, maxtr=2*maxst)
*          'nst' - # to-ek, 'r' - ix koopdihaty, 'w' - pogpe[hocti.
           common /points/ nst0,r(4,maxst)
           common /blnk/ iord(maxst),list(0:maxtr),jord(maxst)

           ntg=1
           nd=nst-nst0
           do 10 np=1,nd
              call searin(ntg,np+nst0)
              jord(np)=ntg
  10          continue
           call kb07ai(jord,nd,iord)
           nt=ntr
           list(nt)=nst
           do 20 i3=nst,nst0+1,-1
              ns=i3-nst0
              iord(i3)=iord(ns)+nst0
1010          if(jord(ns).eq.nt) goto 1020
                 nt=nt-1
                 if(nt.lt.0) then
                    print *,' abend in ''prepar'''
                    stop 21
                    end if
                 list(nt)=i3
                 goto 1010
1020          continue
  20          continue
           do 30 k0=0,nt-1
              list(k0)=nst0
  30          continue
           do 40 i4=1,nst0
  40          iord(i4)=i4
           return
           end

           subroutine addpts(ntr1,nst1)
*
*          ------------------------------
*
                parameter(minst=70000,maxst=minst, maxtr=2*maxst)
           integer next,sites,st,sh
           common /points/ nst,r(4,maxst)
           common /net/ ntr,next(3,maxtr),sites(3,maxtr)
           common /repers/ e(3,3,minst)
           common  /blnk/iord(maxst),list(0:maxtr),jord(maxst)
           real h2(3),q(3)
           real a(4,3),c(6,3)
           sh(m)=mod(m,3)+1
           save

           ns=list(0)
           do 50 nt=1,ntr
              nds=list(nt)-ns
              if(nds.gt.0) then
                 call affin(nt,a,c)
                 nsct=nt
                 do 40 k0=1,nds
                    np=iord(k0+ns)
                    call searin(nsct,np)
                    if(nsct.ne.nt) then
                       print *,'bep[iha',ns+k0,' he e tpeugolxhiku',nsct
```

```
               end if
            call wghts(h2,nt,np)
c           ihteppolqciq: bzbe(ehhoe cpedhee ot tpex bep(ih
c           h2(1)+h2(2)+h2(3)=1
            rad=0
            do 30 nv=1,3
               st=sites(nv,nt)
               do 20 m1=2,3
                  q(m1)=0
                  do 10 m2=1,3
                     q(m1)=q(m1)+e(m1,m2,st)*r(m2,np)
10                   continue
20                continue
               t0=asin(sqrt(q(2)**2+q(3)**2))/sqrt(q(2)**2+q(3)**2)
               q(2)=q(2)*t0
               q(3)=q(3)*t0
               x=a(1,nv)*q(2)+a(3,nv)*q(3)
               y=a(2,nv)*q(2)+a(4,nv)*q(3)
               if(x.lt.0) then
                  print *,' error: x=',x
                  end if
               if(y.lt.0) then
                  print *,' error: y=',y
                  end if
               w=r(4,st)+(c(1,nv)+c(3,nv)*x)*x+(c(2,nv)+c(4,nv)*y)*y
     .            +x*y*((c(5,nv)+c(6,nv))/2+
     .            (c(5,nv)-c(6,nv))/2*(x-y)/sqrt(x**2+y**2))
               rad=rad+w*h2(nv)
30             continue
            r(4,np)=rad
40          continue
         end if
      ns=list(nt)
50    continue
   return
   end

   subroutine affin(nt,a,c)
c  ------------------------

      parameter(minst=70000,maxst=minst, maxtr=2*maxst)
   integer next,sites,st1,st2,st3,sh
   common /points/ nst,r(4,maxst)
   common /net/ ntr,next(3,maxtr),sites(3,maxtr)
   common /repers/ e(3,3,minst),tang(2:3,minst)
   real a(4,3),c(6,3),dx(2),dy(2),e2(3,3),e3(3,3),dv2(3),dv3(3)
   real*8 det
   sh(m)=mod(m,3)+1
   save

   do 80 m1=1,3
      m2=sh(m1)
      m3=sh(m2)
      st1=sites(m1,nt)
      st2=sites(m2,nt)
      st3=sites(m3,nt)
      do 30 i1=2,3
         do 20 i0=1,3
            e2(i0,i1)=0
            e3(i0,i1)=0
            do 10 k0=1,3
               e2(i0,i1)=e2(i0,i1)+e(i0,k0,st1)*e(i1,k0,st2)
               e3(i0,i1)=e3(i0,i1)+e(i0,k0,st1)*e(i1,k0,st3)
10             continue
20          continue
30       continue
      dx1=0
      dy1=0
      dz1=0
      dx2=0
      dy2=0
      dz2=0
      do 40 j0=1,3
         dx1=dx1+e(2,j0,st1)*r(j0,st2)
         dy1=dy1+e(3,j0,st1)*r(j0,st2)
         dz1=dz1+e(1,j0,st1)*r(j0,st2)
         dx2=dx2+e(2,j0,st1)*r(j0,st3)
         dy2=dy2+e(3,j0,st1)*r(j0,st3)
         dz2=dz2+e(1,j0,st1)*r(j0,st3)
40       continue
      s1=sqrt(dx1**2+dy1**2)
      t1=asin(s1)/s1
      dx1=dx1*t1
      dy1=dy1*t1
      s2=sqrt(dx2**2+dy2**2)
      t2=asin(s2)/s2
      dx2=dx2*t2
      dy2=dy2*t2
      do 70 j1=1,3
         dv2(j1)=0
         dv3(j1)=0
         do 60 j2=2,3
            dv2(j1)=dv2(j1)+e2(j1,j2)*tang(j2,st2)
            dv3(j1)=dv3(j1)+e3(j1,j2)*tang(j2,st3)
60          continue
70       continue
      det=dprod(dx1,dy2)-dprod(dx2,dy1)
      a(1,m1)=dy2/det
      a(2,m1)=-dy1/det
      a(3,m1)=-dx2/det
      a(4,m1)=dx1/det
      t2st1=tang(2,st1)
      t3st1=tang(3,st1)
      d1=dx1*t2st1+dy1*t3st1
      d2=dx2*t2st1+dy2*t3st1
      dvx1=(dx1*dv2(2)+dy1*dv2(3))*dz1-dv2(1)*t1*s1**2
      dvy2=(dx2*dv3(2)+dy2*dv3(3))*dz2-dv3(1)*t2*s2**2
      c(1,m1)=d1
      c(2,m1)=d2
      c(3,m1)=2*(r(4,st2)-r(4,st1))-dvx1/2-d1*3/2
      c(4,m1)=2*(r(4,st3)-r(4,st1))-dvy2/2-d2*3/2
      rhs1=(-dv2(2)*dy1+dv2(3)*dx1)*s1+dy1*t2st1-dx1*t3st1
      rhs2=(dv3(2)*dy2-dv3(3)*dx2)*s2-dy2*t2st1+dx2*t3st1
      sc=dy1*dy2+dx1*dx2
      c(5,m1)=(rhs1*det+2*c(3,m1)*sc)/s1**2
      c(6,m1)=(rhs2*det+2*c(4,m1)*sc)/s2**2
80    continue
   return
   end

   subroutine wghts(h,nt,np)
c  -------------------------
c  h2 - ploladi**2 tpex obpazobab(ixcq tpeugolxhikob.
c  h2 = ((x,y]/2)**2
      parameter(minst=70000,maxst=minst, maxtr=2*maxst)
   parameter (eps=1e-6)
   integer next,sites,st1,st2,st3,sh
   common /points/ nst,r(4,maxst)
```

```
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      real h(3),rp(3),tgn(3)
      sh(m)=mod(m,3)+1
      save

      st1=sites(1,nt)
      st2=sites(2,nt)
      st3=sites(3,nt)
      do 10 i0=1,3
         tgn(i0)=
     .    (r(sh(i0),st1)-r(sh(i0),st3))*(r(sh(i0+1),st2)-r(sh(i0+1),st3))
     .   -(r(sh(i0+1),st1)-r(sh(i0+1),st3))*(r(sh(i0),st2)-r(sh(i0),st3))
10    continue
      r1=0
      r2=0
      r3=0
      do 20 m1=1,3
         r1=r1+(r(m1,np)-r(m1,st1))**2
         r2=r2+(r(m1,np)-r(m1,st2))**2
         r3=r3+(r(m1,np)-r(m1,st3))**2
20    continue
      if(r1.lt.eps) then
         h(1)=1
         h(2)=0
         h(3)=0
      else if(r2.lt.eps) then
         h(1)=0
         h(2)=1
         h(3)=0
      else if(r3.lt.eps) then
         h(1)=0
         h(2)=0
         h(3)=1
      else
         sc1=0
         sc2=0
         do 30 i1=1,3
            sc1=sc1+r(i1,np)*tgn(i1)
            sc2=sc2+r(i1,st1)*tgn(i1)
30       continue
         do 40 i2=1,3
            rp(i2)=r(i2,np)*sc2/sc1
40       continue
         h1=area2(rp,r(1,st2),r(1,st3))
         h2=area2(rp,r(1,st3),r(1,st1))
         h3=area2(rp,r(1,st1),r(1,st2))
         h(1)=(h2*r2**2+h3*r3**2)*h1
         h(2)=(h3*r3**2+h1*r1**2)*h2
         h(3)=(h1*r1**2+h2*r2**2)*h3
         z=h(1)+h(2)+h(3)
         h(1)=h(1)/z
         h(2)=h(2)/z
         h(3)=h(3)/z
         end if
      return
      if(h(1).lt.h(2)) then
         him=h2
         im=2
         h(1)=0
      else
         him=h1
         im=1
         h(2)=0
         end if
      if(h(3).lt.h(im)) then
         h(3)=0
         h(im)=1
      else
         h(3)=1
         h(im)=0
         end if
      return
      end

      subroutine searin(tgl,np)
*     _________________________
*     poick tpeugolxhika b kotopyj popadaet to-ka np
            parameter(minst=70000,maxst=minst, maxtr=2*maxst)
      integer next,sites,tgl,m,np,prev
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      save

      mk=0
      a3=volum(r(1,np),r(1,sites(1,tgl)),r(1,sites(2,tgl)))
      a1=volum(r(1,np),r(1,sites(2,tgl)),r(1,sites(3,tgl)))
      a2=volum(r(1,np),r(1,sites(3,tgl)),r(1,sites(1,tgl)))
      if(a1.ge.0.and.a2.ge.0.and.a3.ge.0) then
          return
          end if
      if(a2.lt.a3) then
          if(a1.lt.a2) then
              m=1
          else
              m=2
              end if
      else
          if(a1.lt.a3) then
              m=1
          else
              m=3
              end if
          end if
      k0=0
1010  prev=tgl
      tgl=next(m,tgl)
      m2=mark(tgl,prev)
      m3=mod(m2,3)+1
      m1=mod(m3,3)+1
      a3=volum(r(1,np),r(1,sites(m1,tgl)),r(1,sites(m2,tgl)))
      a1=volum(r(1,np),r(1,sites(m2,tgl)),r(1,sites(m3,tgl)))
      if(a1.lt.a3) then
          m=m1
          a=a1
      else
          m=m3
          a=a3
          end if
      k0=k0+1
      if(k0.gt.maxtr) then
          print *,' ===> tpeugolxhik he hajden <===='
          stop 31
          end if
      if(a.lt.0) goto 1010
      return
      end
```

ints.f 3

```
            end if
         call wghts(h2,nt,np)
*        ihteppolqciq: bzbelehhoe cpedhee ot tpex bep(ih
*        h2(1)+h2(2)+h2(3)=1
         rad=0
         do 30 nv=1,3
            st=sites(nv,nt)
            do 20 m1=2,3
               q(m1)=0
               do 10 m2=1,3
                  q(m1)=q(m1)+e(m1,m2,st)*r(m2,np)
10                continue
20             continue
            t0=asin(sqrt(q(2)**2+q(3)**2))/sqrt(q(2)**2+q(3)**2)
            q(2)=q(2)*t0
            q(3)=q(3)*t0
            x=a(1,nv)*q(2)+a(3,nv)*q(3)
            y=a(2,nv)*q(2)+a(4,nv)*q(3)
            if(x.lt.0) then
               print *,' error: x=',x
               end if
            if(y.lt.0) then
               print *,' error: y=',y
               end if
            w=r(4,st)+(c(1,nv)+c(3,nv)*x)*x+(c(2,nv)+c(4,nv)*y)*y
     *         +x*y*((c(5,nv)+c(6,nv))/2+
     *         (c(5,nv)-c(6,nv))/2*(x-y)/sqrt(x**2+y**2))
            rad=rad+w*h2(nv)
30          continue
         r(4,np)=rad
40       continue
      end if
   ns=list(nt)
50 continue
   return
   end

   subroutine affin
*  ___________
*
```

```
20          continue
30       continue
      dx1=0
      dy1=0
      dz1=0
      dx2=0
      dy2=0
      dz2=0
      do 40 j0=1,3
         dx1=dx1+e(2,j0,st1)*r(j0,st2)
         dy1=dy1+e(3,j0,st1)*r(j0,st2)
         dz1=dz1+e(1,j0,st1)*r(j0,st2)
         dx2=dx2+e(2,j0,st1)*r(j0,st3)
         dy2=dy2+e(3,j0,st1)*r(j0,st3)
         dz2=dz2+e(1,j0,st1)*r(j0,st3)
40       continue
      s1=sqrt(dx1**2+dy1**2)
      t1=asin(s1)/s1
      dx1=dx1*t1
      dy1=dy1*t1
      s2=sqrt(dx2**2+dy2*
      t2=asin(s2)/
      dx2=dx2
```

4

```
,next(3,maxtr),sites(3,maxtr)
,tgn(3)

t)
nt)
,nt)
3

0),st1)-r(sh(10),st3))*(r(sh(10+1),st2)-r(sh(10+1),st3))
.0+1),st1)-r(sh(10+1),st3))*(r(sh(10),st2)-r(sh(10),st3))
ue

np)-r(m1,st1))**2
(m1,st2))**2
))**2
```

```
   if(h(3).lt.h(1m)) then
      h(3)=0
      h(1m)=1
   else
      h(3)=1
      h(1m)=0
      end if
   return
   end

   subroutine searin(tg1,np)
*  poick tpeugolxhika b kotopyj popadaet to-ka np
*     parameter(minst=70000,maxst=minst, maxtr=2*maxst)
   integer next,sites,tg1,m,np,prev
   common /points/ nst,r(4,maxst)
   common /net/ ntr,next(3,maxtr),sites(3,maxtr)
   save

   mk=0
   a3=volum(r(1,np),r(1,sites(1,tg1)),r(1,sites(2,tg1)))
   a1=volum(r(1,np),r(1,sites(2,tg1)),r(1,sites(3,tg1)))
   a2=volum(r(1,np),r(1,sites(3,tg1)),r(1,sites(1,tg1)))
   if(a1.ge.0.and.a2.ge.0.and.a3.ge.0) then
      return
      end if
   if(a2.lt.a3) then
      if(a1.lt.a2) then
         m=1
      else
         m=2
         end if
```

```
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      real h(3),rp(3),tgn(3)
      sh(m)=mod(m,3)+1
      save

      st1=sites(1,nt)
      st2=sites(2,nt)
      st3=sites(3,nt)
      do 10 i0=1,3
         tgn(i0)=
     .   (r(sh(i0),st1)-r(sh(i0),st3))*(r(sh(i0+1),st2)-r(sh(i0+1),st3))
     .  -(r(sh(i0+1),st1)-r(sh(i0+1),st3))*(r(sh(i0),st2)-r(sh(i0),st3))
10    continue
      r1=0
      r2=0
      r3=0
      do 20 m1=1,3
         r1=r1+(r(m1,np)-r(m1,st1))**2
         r2=r2+(r(m1,np)-r(m1,st2))**2
         r3=r3+(r(m1,np)-r(m1,st3))**2
20    continue
      if(r1.lt.eps) then
         h(1)=1
         h(2)=0
         h(3)=0
      else if(r2.lt.eps) then
         h(1)=0
         h(2)=1
         h(3)=0
      else if(r3.lt.eps) then
         h(1)=0
         h(2)=0
         h(3)=1
      else
         sc1=0
         sc2=0
         do 30 i1=1,3
            sc1=sc1+r(i1,np)*tgn(i1)
            sc2=sc2+r(i1,st1)*tgn(i1)
30       continue
         do 40 i2=1,3
            rp(i2)=r(i2,np)*sc2/sc1
40       continue
         h1=area2(rp,r(1,st2),r(1,st3))
         h2=area2(rp,r(1,st3),r(1,st1))
         h3=area2(rp,r(1,st1),r(1,st2))
         h(1)=(h2*r2**2+h3*r3**2)*h1
         h(2)=(h3*r3**2+h1*r1**2)*h2
         h(3)=(h1*r1**2+h2*r2**2)*h3
         z=h(1)+h(2)+h(3)
         h(1)=h(1)/z
         h(2)=h(2)/z
         h(3)=h(3)/z
         end if
      return
      if(h(1).lt.h(2)) then
         him=h2
         im=2
         h(1)=0
      else
         him=h1
         im=1
         h(2)=0
         end if
      if(h(3).lt.h(im)) then
         h(3)=0
         h(im)=1
      else
         h(3)=1
         h(im)=0
         end if
      return
      end

      subroutine searin(tgl,np)
*     _________________________
*     polck tpeugolxhika b kotopyj popadaet to~ka np
            parameter(minst=70000,maxst=minst, maxtr=2*maxst)
      integer next,sites,tgl,m,np,prev
      common /points/ nst,r(4,maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      save

      mk=0
      a3=volum(r(1,np),r(1,sites(1,tgl)),r(1,sites(2,tgl)))
      a1=volum(r(1,np),r(1,sites(2,tgl)),r(1,sites(3,tgl)))
      a2=volum(r(1,np),r(1,sites(3,tgl)),r(1,sites(1,tgl)))
      if(a1.ge.0.and.a2.ge.0.and.a3.ge.0) then
          return
          end if
      if(a2.lt.a3) then
          if(a1.lt.a2) then
              m=1
          else
              m=2
              end if
      else
          if(a1.lt.a3) then
              m=1
          else
              m=3
              end if
          end if
      k0=0
1010  prev=tgl
      tgl=next(m,tgl)
      m2=mark(tgl,prev)
      m3=mod(m2,3)+1
      m1=mod(m3,3)+1
      a3=volum(r(1,np),r(1,sites(m1,tgl)),r(1,sites(m2,tgl)))
      a1=volum(r(1,np),r(1,sites(m2,tgl)),r(1,sites(m3,tgl)))
      if(a1.lt.a3) then
          m=m1
          a=a1
      else
          m=m3
          a=a3
          end if
      k0=k0+1
      if(k0.gt.maxtr) then
          print *,' ===> tpeugolxhik he hajdeh <==='
          stop 31
          end if
      if(a.lt.0) goto 1010
      return
      end
```

```
      real function area2(x,y,z)
*     _____________________________
*     plo}adx tpeugolxhlka b kbadrate
*     area2 = (x,y)**2/4
      real x(3),y(3),z(3)
      area2=0
      do 10 m=1,3
        n=mod(m,3)+1
        area2=area2+((x(m)-z(m))*(y(n)-z(n))-(x(n)-z(n))*(y(m)-z(m)))**2
 10   continue
      area2=area2/4
      return
      end


      subroutine outint(ntr1,nst1)
*     ____________________________
*     bybod dlq bizualizacii
            parameter(minst=70000,maxst=minst, maxtr=2*maxst)
      integer sites
      common /points/ nst,r(4,maxst)
      common /repers/ mord(maxst)
      common  /blnk/iord(maxst),list(0:maxtr),jord(maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      save

      write (7,*) ntr1,nst1
      do 10 j2=1,nst1
          write (7,*) ((r(i1,iord(j2))*r(4,iord(j2))),i1=1,3)
 10     continue
      do 20 i2=1,nst1
         mord(iord(i2))=i2
 20     continue
      read (2) ((sites(i1,j2),i1=1,3),j2=1,ntr1)
      do 11 j2=1,ntr1
      write (7,*) (mord(sites(i1,j2)),i1=1,3)
 11     continue
      close(7)
      return
      end



      subroutine outint1(ntr1,nst1)
*     _____________________________
*     bybod dlq bizualizacii
            parameter(minst=70000,maxst=minst, maxtr=2*maxst)
      integer sites
      common /points/ nst,r(4,maxst)
      common /repers/ mord(maxst)
      common  /blnk/iord(maxst),list(0:maxtr),jord(maxst)
      common /net/ ntr,next(3,maxtr),sites(3,maxtr)
      save
      read (2) ((sites(i1,j2),i1=1,3),j2=1,ntr1)
      nt = 0
      do 712 it = 1,ntr1
      if(r(4,sites(1,it)).gt. 1.5) goto 712
      if(r(4,sites(2,it)).gt. 1.5) goto 712
      if(r(4,sites(3,it)).gt. 1.5) goto 712
      nt = nt+1
712   continue

      write (7,*) nt,nst1
      do 10 j2=1,nst1
          write (7,*) ((r(i1,iord(j2))*r(4,iord(j2))),i1=1,3)
 10     continue
      do 20 i2=1,nst1
         mord(iord(i2))=i2
 20     continue
      do 11 j2=1,ntr1
      if(r(4,sites(1,j2)).gt. 1.5) goto 11
      if(r(4,sites(2,j2)).gt. 1.5) goto 11
      if(r(4,sites(3,j2)).gt. 1.5) goto 11
      write (7,*) (mord(sites(i1,j2)),i1=1,3)
 11     continue
      close(7)
      return
      end
```

```
c1       kb07a          21/04/80                                        kb000010
cname kb07a(r)                                                          kb000020
      subroutine kb07a(count,n,index)                                   kb000030
c                                                                       kb000040
c             kb07a      handles real single-length variables         kb000050
c standard fortran 66 (a verified pfort subroutine)                     kb000060
c  the work-space 'mark' of length 50 permits up to 2**(50/2) numbers   kb000070
c  to be sorted. this is more than the ibm virtual memory space         kb000080
c  will hold .                                                          kb000090
      dimension count(n),mark(50),index(n)                              kb000100
c  set index array to original order .                                  kb000110
      do 10 i=1,n                                                       kb000120
      index(i)=i                                                        kb000130
   10 continue                                                          kb000140
c  check that a trivial case has not been entered .                     kb000150
      if(n.eq.1)goto 200                                                kb000160
      if(n.ge.1)go to 30                                                kb000170
      write(6,20)                                                       kb000180
   20 format(///20x,65h ***kb07a*** no numbers to be sorted ** return tokb000190
     2 calling program )                                                kb000200
      goto 200                                                          kb000210
c  'm' is the length of segment which is short enough to enter          kb000220
c  the final sorting routine. it may be easily changed.                 kb000230
   30 m=12                                                              kb000240
c  set up initial values.                                               kb000250
      la=2                                                              kb000260
      is=1                                                              kb000270
      if=n                                                              kb000280
      do 190 mloop=1,n                                                  kb000290
c  if segment is short enough sort with final sorting routine .         kb000300
      ifka=if-is                                                        kb000310
      if((ifka+1).gt.m)goto 70                                          kb000320
c********** final sorting ***                                          kb000330
   ( a simple bubble sort )                                             kb000340
      isl=is+1                                                          kb000350
      do 60 j=isl,if                                                    kb000360
      i=j                                                               kb000370
   40 if(count(i-1).lt.count(i))goto 60                                 kb000380
      if(count(i-1).gt.count(i))goto 50                                 kb000390
      if(index(i-1).lt.index(i))goto 60                                 kb000400
   50 av=count(i-1)                                                     kb000410
      count(i-1)=count(i)                                               kb000420
      count(i)=av                                                       kb000430
      int=index(i-1)                                                    kb000440
      index(i-1)=index(i)                                               kb000450
      index(i)=int                                                      kb000460
      i=i-1                                                             kb000470
      if(i.gt.is)goto  40                                               kb000480
   60 continue                                                          kb000490
      la=la-2                                                           kb000500
      goto 170                                                          kb000510
            ******* quicksort ********                                  kb000520
   select the number in the central position in the segment as          kb000530
   the test number.replace it with the number from the segment's        kb000540
   highest address.                                                     kb000550
   70 iy=(is+if)/2                                                      kb000560
      x=count(iy)                                                       kb000570
      intest=index(iy)                                                  kb000580
      count(iy)=count(if)                                               kb000590
      index(iy)=index(if)                                               kb000600
   the markers 'i' and 'ifk' are used for the beginning and end         kb000610
   of the section not so far tested against the present value           kb000620
   of x .                                                               kb000630
      k=1                                                               kb000640
      ifk=if                                                            kb000650
c  we alternate between the outer loop that increases i and the         kb000660
c  inner loop that reduces ifk, moving numbers and indices as           kb000670
c  necessary, until they meet .                                         kb000680
      do 110 i=is,if                                                    kb000690
      if(x.gt.count(i))goto 110                                         kb000700
      if(x.lt.count(i))goto 80                                          kb000710
      if(intest.gt.index(i))goto 110                                    kb000720
   80 if(i.ge.ifk)goto 120                                              kb000730
      count(ifk)=count(i)                                               kb000740
      index(ifk)=index(i)                                               kb000750
      k1=k                                                              kb000760
      do 100 k=k1,ifka                                                  kb000770
      ifk=if-k                                                          kb000780
      if(count(ifk).gt.x)goto 100                                       kb000790
      if(count(ifk).lt.x)goto 90                                        kb000800
      if(intest.le.index(ifk))goto 100                                  kb000810
   90 if(i.ge.ifk)goto 130                                              kb000820
      count(i)=count(ifk)                                               kb000830
      index(i)=index(ifk)                                               kb000840
      go to 110                                                         kb000850
  100 continue                                                          kb000860
      goto 120                                                          kb000870
  110 continue                                                          kb000880
c  return the test number to the position marked by the marker          kb000890
c  which did not move last. it divides the initial segment into         kb000900
c  2 parts. any element in the first part is less than or equal         kb000910
c  to any element in the second part, and they may now be sorted        kb000920
c  independently .                                                      kb000930
  120 count(ifk)=x                                                      kb000940
      index(ifk)=intest                                                 kb000950
      ip=ifk                                                            kb000960
      goto 140                                                          kb000970
  130 count(i)=x                                                        kb000980
      index(i)=intest                                                   kb000990
      ip=i                                                              kb001000
c  store the longer subdivision in workspace.                           kb001010
  140 if((ip-is).gt.(if-ip))goto 150                                    kb001020
      mark(la)=if                                                       kb001030
      mark(la-1)=ip+1                                                   kb001040
      if=ip-1                                                           kb001050
      goto 160                                                          kb001060
  150 mark(la)=ip-1                                                     kb001070
      mark(la-1)=is                                                     kb001080
      is=ip+1                                                           kb001090
c  find the length of the shorter subdivision.                          kb001100
  160 lngth=if-is                                                       kb001110
      if(lngth.le.0)goto 180                                            kb001120
c  if it contains more than one element supply it with workspace .      kb001130
      la=la+2                                                           kb001140
      goto 190                                                          kb001150
  170 if(la.le.0)goto 200                                               kb001160
c  obtain the address of the shortest segment awaiting quicksort        kb001170
  180 if=mark(la)                                                       kb001180
      is=mark(la-1)                                                     kb001190
  190 continue                                                          kb001200
  200 return                                                            kb001210
      end                                                               kb001220
```

```
      subroutine kb07ai(count,n,index)                                  kb000030
c                                                                       kb000040
c             kb07a      handles real single-length variables         kb000050
c standard fortran 66 (a verified pfort subroutine)                     kb000060
c  the work-space 'mark' of length 50 permits up to 2**(50/2) numbers   kb000070
c  to be sorted. this is more than the ibm virtual memory space        kb000080
c  will hold .                                                          kb000090
      implicit integer (a-z)
      integer count(n),mark(50),index(n)
c  set index array to original order .                                  kb000110
      do 10 i=1,n                                                       kb000120
      index(i)=i                                                        kb000130
   10 continue                                                          kb000140
c  check that a trivial case has not been entered .                     kb000150
      if(n.eq.1)goto 200                                                kb000160
      if(n.ge.1)go to 30                                                kb000170
      write(6,20)                                                       kb000180
   20 format(///20x,65h ***kb07a*** no numbers to be sorted ** return tokb000190
     2 calling program )                                                kb000200
      goto 200                                                          kb000210
c  'm' is the length of segment which is short enough to enter          kb000220
c  the final sorting routine. it may be easily changed.                 kb000230
   30 m=12                                                              kb000240
c  set up initial values.                                               kb000250
      ia=2                                                              kb000260
      is=1                                                              kb000270
      if=n                                                              kb000280
      do 190 mloop=1,n                                                  kb000290
c  if segment is short enough sort with final sorting routine .         kb000300
      ifka=if-is                                                        kb000310
      if((ifka+1).gt.m)goto 70                                          kb000320
c********* final sorting ***                                           kb000330
c  ( a simple bubble sort )                                             kb000340
      isi=is+1                                                          kb000350
      do 60 j=isi,if                                                    kb000360
      i=j                                                               kb000370
   40 if(count(i-1).lt.count(i))goto 60                                 kb000380
      if(count(i-1).gt.count(i))goto 50                                 kb000390
      if(index(i-1).lt.index(i))goto 60                                 kb000400
   50 av=count(i-1)                                                     kb000410
      count(i-1)=count(i)                                               kb000420
      count(i)=av                                                       kb000430
      int=index(i-1)                                                    kb000440
      index(i-1)=index(i)                                               kb000450
      index(i)=int                                                      kb000460
      i=i-1                                                             kb000470
      if(i.gt.is)goto  40                                               kb000480
   60 continue                                                          kb000490
      ia=ia-2                                                           kb000500
      goto 170                                                          kb000510
            ******* quicksort ********                                  kb000520
   select the number in the central position in the segment as          kb000530
   the test number.replace it with the number from the segment's        kb000540
   highest address.                                                     kb000550
   70 iy=(is+if)/2                                                      kb000560
      x=count(iy)                                                       kb000570
      intest=index(iy)                                                  kb000580
      count(iy)=count(if)                                               kb000590
      index(iy)=index(if)                                               kb000600
   the markers 'i' and 'ifk' are used for the beginning and end         kb000610
   of the section not so far tested against the present value           kb000620
   of x .                                                               kb000630
      k=1                                                               kb000640
      ifk=if                                                            kb000650
c  we alternate between the outer loop that increases i and the         kb000660
c  inner loop that reduces ifk, moving numbers and indices as           kb000670
c  necessary, until they meet .                                         kb000680
      do 110 i=is,if                                                    kb000690
      if(x.gt.count(i))goto 110                                         kb000700
      if(x.lt.count(i))goto 80                                          kb000710
      if(intest.gt.index(i))goto 110                                    kb000720
   80 if(i.ge.ifk)goto 120                                              kb000730
      count(ifk)=count(i)                                               kb000740
      index(ifk)=index(i)                                               kb000750
      k1=k                                                              kb000760
      do 100 k=k1,ifka                                                  kb000770
      ifk=if-k                                                          kb000780
      if(count(ifk).gt.x)goto 100                                       kb000790
      if(count(ifk).lt.x)goto 90                                        kb000800
      if(intest.le.index(ifk))goto 100                                  kb000810
   90 if(i.ge.ifk)goto 130                                              kb000820
      count(i)=count(ifk)                                               kb000830
      index(i)=index(ifk)                                               kb000840
      go to 110                                                         kb000850
  100 continue                                                          kb000860
      goto 120                                                          kb000870
  110 continue                                                          kb000880
c  return the test number to the position marked by the marker          kb000890
c  which did not move last. it divides the initial segment into         kb000900
c  2 parts. any element in the first part is less than or equal         kb000910
c  to any element in the second part, and they may now be sorted        kb000920
c  independently .                                                      kb000930
  120 count(ifk)=x                                                      kb000940
      index(ifk)=intest                                                 kb000950
      ip=ifk                                                            kb000960
      goto 140                                                          kb000970
  130 count(i)=x                                                        kb000980
      index(i)=intest                                                   kb000990
      ip=i                                                              kb001000
c  store the longer subdivision in workspace.                           kb001010
  140 if((ip-is).gt.(if-ip))goto 150                                    kb001020
      mark(ia)=if                                                       kb001030
      mark(ia-1)=ip+1                                                   kb001040
      if=ip-1                                                           kb001050
      goto 160                                                          kb001060
  150 mark(ia)=ip-1                                                     kb001070
      mark(ia-1)=is                                                     kb001080
      is=ip+1                                                           kb001090
c  find the length of the shorter subdivision.                          kb001100
  160 lngth=if-is                                                       kb001110
      if(lngth.le.0)goto 180                                            kb001120
c  if it contains more than one element supply it with workspace .      kb001130
      ia=ia+2                                                           kb001140
      goto 190                                                          kb001150
  170 if(ia.le.0)goto 200                                               kb001160
c  obtain the address of the shortest segment awaiting quicksort        kb001170
  180 if=mark(ia)                                                       kb001180
      is=mark(ia-1)                                                     kb001190
  190 continue                                                          kb001200
  200 return                                                            kb001210
      end                                                               kb001220
```

```
      subroutine plider(name)
      character*8 name
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      save
      open(4,file='leader')
      write(4,*) '     ****************************************'
      write(4,*) '     ***** information about the execution *****'
      write(4,2) name
    2 format(1x, '     ***** the program ',a8,  '  has run   *****')
      write(4,*) '     ****************************************'
      write(4,*) ' is the input file formatted or unformatted ?(f/u)'
      if(ifo.ne.0) write(4,1)  'f'
      if(ifo.eq.0) write(4,1)  'u'
      write(4,*)
     &  ' does the input file contain errors ?(y/n)'
      if(ipog.ne.0) write(4,1)  'y'
      if(ipog.eq.0) write(4,1)  'n'
      write(4,*)
     &' is this surface or function with several values ?(s/f)'
      if(isu.eq.0) write(4,1)  'f'
      if(isu.ne.0) write(4,1)  's'
      if(isu.ne.0) goto 20
      write(4,*) ' how many values has the function ?   '
      write(4,*) nmi0
   20 write(4,*) ' minimal and maximal values of  x          '
      write(4,*) xmin,xmax
      write(4,*) ' minimal and maximal values of  y          '
      write(4,*) ymin,ymax
      write(4,*) ' minimal and maximal values of  z          '
      write(4,*) zmin,zmax
      write(4,*) ' the size of grid for averaging'
      write(4,*) grid
      if(isu.eq.0) goto 30
      write(4,*) ' the maximal number of points in the model'
      write(4,*) madst
   30 write(4,*) ' the radius of confidence'
      write(4,*) rdmax0
      write(4,*) ' what smoothing do you prefer ? (n/1/2/q)'
      write(4,*) '       n - no smoothing  '
      write(4,*) '       1 - smooth according to errors (from the file)'
      write(4,*) '       2 - smooth (all errors = 1) '
      write(4,*) '       q - request during the execution'
      if(kva .eq.0) write(4,1)  'n'
      if(kva .eq.1) write(4,1)  '1'
      if(kva .eq.2) write(4,1)  '2'
      if(kva .lt.0) write(4,1)  'q'
      write(4,*) ' kill the peaks ?  (n/1/2/q):'
      write(4,*) '       n - no'
      write(4,*) '       1 - kill according to errors (from the file)'
      write(4,*) '       2 - kill by statistical method'
      write(4,*) '       q - request during the execution'
      if(ipik.eq.0) write(4,1)  'n'
      if(ipik.eq.1) write(4,1)  '1'
      if(ipik.eq.2) write(4,1)  '2'
      if(ipik.lt.0) write(4,1)  'q'
      write(4,*) ' draw the triangulation during the execution? (y/n/q)'
      if(idra.eq.0) write(4,1)  'n'
      if(idra.gt.0) write(4,1)  'y'
      if(idra.lt.0) write(4,1)  'q'
      if(isu.ne.0) goto 77
      write(4,*) ' will you work with windows ?'
      if(iwin.eq.0) write(4,1)  'n'
      if(iwin.ne.0) write(4,1)  'y'
      write(4,*) ' draw the maps during the execution ? (y/r/q)'
      if(idrm.eq.0) write(4,1)  'n'
      if(idrm.gt.0) write(4,1)  'y'
      if(idrm.lt.0) write(4,1)  'q'
      write(4,*) ' in what form store the maps   ?'
      write(4,*) '       s - standart '
      write(4,*) '       b - bytes  '
      write(4,*) '       a - both (standart and bytes) '
      write(4,*) '       n - don"t store'
      if(msav.eq.1) write(4,1)  's'
      if(msav.eq.2) write(4,1)  'b'
      if(msav.eq.3) write(4,1)  'a'
      if(msav.eq.0) write(4,1)  'n'
      write(4,*) ' take the logarithm of input data ?'
      write(4,*) '       0 - don"t take logarithm'
      write(4,*) '       1 - take logarithm (once)'
      write(4,*) '       2 - take logarithm (twice) etc.'
      write(4,*)  ilg
      write(4,*) ' the number of mineral under investigation'
      write(4,*) '( if 0 - requst during the execution )'
      write(4,*)  miner
   77 continue
      close(4,*)
    1 format(a1)
      return
      end

      subroutine glider
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      save
      character*1 r
      open(4,file='leader')
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,1)  r
      ifo=-1
      if(r.eq.'f') ifo=1
      if(r.eq.'u') ifo=0
      if(ifo.lt.0) goto 227
      read(4,*)
      read(4,1)  r
      ipog=-1
      if(r.eq.'y') ipog=1
      if(r.eq.'n') ipog=0
      if(ipog.lt.0) goto 227
      read(4,*)
      read(4,1)  r
      isu=-1
      if(r.eq.'s') isu=1
      if(r.eq.'f') isu=0
      nmi0=1
      if(isu.eq.1) goto 20
      read(4,*)
      read(4,*) nmi0
   20 read(4,*)
      read(4,*) xmin,xmax
      read(4,*)
      read(4,*) ymin,ymax
```

```
      read(4,*)
      read(4,*) zmin,zmax
      read(4,*)
      read(4,*) grid
      if(isu.eq.0) goto 30
      read(4,*)
      read(4,*) madst
30    read(4,*)
      read(4,*) rdmax0
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,1)  r
      kva =-1
      if(r.eq.'n')  kva=0
      if(r.eq.'1')  kva=1
      if(r.eq.'2')  kva=2
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,1)  r
      ipik =-1
      if(r.eq.'n')  ipik=0
      if(r.eq.'1')  ipik=1
      if(r.eq.'2')  ipik=2
      read(4,*)
      read(4,1)  r
      idra=-1
      if(r.eq.'y') idra=1
      if(r.eq.'n') idra=0
      if(isu.ne.0) goto 77
      read(4,*)
      read(4,1)  r
      iwin=0
      if(r.eq.'y') iwin=1
      read(4,*)
      read(4,1) r
      idrm=-1
      if(r.eq.'y') idrm=1
      if(r.eq.'n') idrm=0
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,1) r
      msav=0
      if(r.eq.'s') msav=1
      if(r.eq.'b') msav=2
      if(r.eq.'a') msav=3
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*)
      read(4,*) ilg
      read(4,*)
      read(4,*)
      read(4,*) miner
77    continue
      close(4,*)
```

```
 1    format(a1)
      return
 227       print *,' errors in file <leader>'
           print *,' fix errors and restart from the beginning'
           stop
      end

      subroutine getans(ans)
      integer*4 ans,suk
      character*1 repl(4)
      equivalence(suk,repl(1))
      suk=0
             print *
             print *,'           <enter> - no'
             print *,'          * <enter> - yes'
      read (5,103,end=1010,err=1010) repl(4)
  103 format (a1)
      if(repl(4).eq.' ') suk=0
      goto 1020
1010  rewind(5)
1020  continue
      ans=suk
      return
      end
*_____________________________________________________________$
      logical function frame( k,r )
        real r(3,1)

***   begin frame
      accy5=  1d-2
      accy6= -1d-2
      frame=r(1,k).lt.-accy5.or.r(1,k).gt.(1+accy5)
      if(frame) return
      frame=r(2,k).lt.-accy5.or.r(2,k).gt.(1+accy5)
      if(frame) return
      frame=r(3,k).lt.accy6
      return
***   end function frame
        end
```

```
cl       mal4a          25/05/79                                        mal100010
cname mal4a(r)                    check                                mal100020
c    mal4a/b/c - a routine to calculate the solution of the general     mal100030
c    linear least squares problem optionally with linear equality       mal100040
c    constraints. the problem is presented as m linear equations in n   mal100050
c    unknowns, the first ml equations defining the constraints.         mal100060
c       additional right hand sides may be passed through a b entry     mal100070
c    which will take advantage of previous work.                        mal100080
c       the variance-covariance matrix and an estimate of the residual  mal100090
c    variance is provided by the c entry.                               mal100100
c       calculation of residuals and provision of printed output are    mal100110
c    are additional options.                                            mal100120
c                                                                       mal100130
      subroutine mal4a (m,n,ml,a,ia,b,x,iw,w,ipr)                       mal100140
      double precision name,nme1,nme2,fm02as                            mal100150
      dimension a(ia,1),b(1),x(1),w(1)                                  mal100160
      dimension v(iv,1),std(1)                                          mal100170
      dimension iw(1)                                                   mal100180
c                                                                       mal100190
c     m    on entry: no. of equations ( including constraints ).        mal100200
c     n    on entry: no. of unknowns.                                   mal100210
c     ml   on entry: no. of constraints.                                mal100220
c     a    on entry: holds the matrix of the equations; on return: holds mal100230
c          the triangularized matrix and transformations; size: m*n.    mal100240
c     ia   on entry: first dimension of a.                              mal100250
c     b    on entry: holds right hand side; on return: transformed right mal100260
c          hand side ( or optionally the residuals ); size: m.          mal100270
c     x    on return: the solution; size: n.                            mal100280
c     iw   on return: column interchanges; size: n.                     mal100290
c     w    on return: transformation information; size: 2*n             mal100300
c     ipr  on entry: specifies output and residuals options.            mal100310
c          ipr even: residuals; ipr odd: no residuals.                  mal100320
c          ipr > 0: no printing; ipr < 0: printing done.                mal100330
c                                                                       mal100340
c     restrictions: m.ge.n; 0.le.ml.le.n.                               mal100350
c                                                                       mal100360
c     reference: bjorck and golub, tech. report cs83, stanford univ.    mal100370
c                                                                       mal100380
c                                    december 1971                      mal100390
c                                                                       mal100400
        private variables                                               mal100410
      logical lswtch,debug,neqml,res,meqn,rths,vflag                    mal100420
      data nme1/' mal4a: '/,nme2/' mal4b: '/                            mal100430
c                                                                       mal100440
        mal4a the initial entry point                                   mal100450
c                                                                       mal100460
      rths=.false.                                                      mal100470
      name=nme1                                                         mal100480
      go to 10                                                          mal100490
c                                                                       mal100500
        mal4b an entry for supplying a new right hand side to a system  mal100510
        of equations solved previously, the variables m,n,ml,a,ia,iw    mal100520
        and w are assumed to have been unchanged.                       mal100530
c                                                                       mal100540
      entry mal4b (b,x,ipr)                                             mal100550
c                                                                       mal100560
c     b    on entry: new right hand side; on return: as above.          mal100570
c     x    on return: new solution; size: n.                            mal100580
c     ipr  as above.                                                    mal100590
c                                                                       mal100600
      name=nme2                                                         mal100610
      rths=.true.                                                       mal100620
      go to 10                                                          mal100630
c                                                                       mal100640
c       entry to obtain variance-covariance matrix                      mal100650
c                                                                       mal100660
      entry mal4c(v,iv,var,std,ipr)                                     mal100670
c                                                                       mal100680
c     v    on return: holds variance-covariance matrix; size: n*n.      mal100690
c     iv   on entry: first dimension of v.                              mal100700
c     var  on return: estimate of residual variance.                    mal100710
c     std  on return: solution standard deviations; size: n.            mal100720
c     ipr  as above.                                                    mal100730
c                                                                       mal100740
      vflag=.true.                                                      mal100750
      go to 540                                                         mal100760
c                                                                       mal100770
c                                                                       mal100780
c       main entry; set switches for printing and evaluating residuals. mal100790
c                                                                       mal100800
c     if ipr even residuals required ( res = true )                     mal100810
   10 res=.false.                                                       mal100820
      if(mod(iabs(ipr),2).eq.0) res=.true.                              mal100830
c       identify special case of m = n                                  mal100840
      meqn=.false.                                                      mal100850
      if(m-n)730,20,30                                                  mal100860
   20 meqn=.true.                                                       mal100870
c       identify special case of n = ml                                 mal100880
   30 neqml=.false.                                                     mal100890
      if(n-ml) 730,40,50                                                mal100900
   40 neqml=.true.                                                      mal100910
   50 if(ml.lt.0) go to 730                                             mal100920
c                                                                       mal100930
      vflag=.false.                                                     mal100940
      if(ipr.ge.0) go to 70                                             mal100950
c                                                                       mal100960
c       print headings etc.                                             mal100970
      mm=m-ml                                                           mal100980
      write(6,60) name,ml,mm,n                                          mal100990
   60 format('0',a8,'constraint equations    = ',i3/                    mal101000
     *       9x,'least squares equations = ',i3/                        mal101010
     *       9x,'number of parameters    = ',i3)                        mal101020
c                                                                       mal101030
c                                                                       mal101040
c       initialize and compute column scale factors                     mal101050
   70 if(rths) go to 320                                                mal101060
      n1=n+1                                                            mal101070
      ml1=ml+1                                                          mal101080
      nn=n-1                                                            mal101090
c     initialize column interchange indicator and scale factors         mal101100
      do 90 j=1,n                                                       mal101110
      w(n+j)=0.                                                         mal101120
   80 iw(j)=j                                                           mal101130
c     calculate column scaling factors.                                 mal101140
      do 110 i=1,m                                                      mal101150
      amx=0.                                                            mal101160
      do 90 j=1,n                                                       mal101170
   90 amx=amax1(abs(a(i,j)),amx)                                        mal101180
      if(amx.eq.0.) go to 110                                           mal101190
      do 100 j=1,n                                                      mal101200
  100 w(n+j)=amax1(w(n+j), abs(a(i,j))/amx)                             mal101210
  110 continue                                                          mal101220
      do 120 j=1,n                                                      mal101230
  120 w(n+j)=1./(w(n+j)*w(n+j))                                         mal101240
      k=1                                                               mal101250
c                                                                       mal101260
c     if ml = 0 bypass first transformation step.                       mal101270
      if(ml.eq.0) go to 290                                             mal101280
```

```
:                                                                       ma101290
.                                                                       ma101300
      begin transformation step q1 on constraint equations.             ma101310
.                                                                       ma101320
      nx=m1                                                             ma101330
      mx=m1                                                             ma101340
      mx1=mx+1                                                          ma101350
                                                                        ma101360
      calculate initial partial column norms and select largest         ma101370
      solution array used as tempory storage space for norms.           ma101380
  130 amx=0.                                                            ma101390
      do 140 j=k,n                                                      ma101400
      x(j)=fm02as(mx1-k,a(k,j),1,a(k,j),1)                              ma101410
      test= abs(x(j))*w(n+j)                                            ma101420
      if(test.le.amx) go to 140                                         ma101430
      jx=j                                                              ma101440
      amx=test                                                          ma101450
  140 continue                                                          ma101460
      go to 170                                                         ma101470
      calculate partial column norms using old values and select largestma101480
  150 k=k+1                                                             ma101490
      amx=0.                                                            ma101500
      do 160 j=k,n                                                      ma101510
      x(j)=x(j)-a(k-1,j)*a(k-1,j)                                       ma101520
      test= abs(x(j))*w(n+j)                                            ma101530
      if(test.le.amx) go to 160                                         ma101540
      jx=j                                                              ma101550
      amx=test                                                          ma101560
  160 continue                                                          ma101570
  170 if(jx.eq.k) go to 190                                             ma101580
      interchange columns so that col. with largest norm is used as pivotma101590
      do 180 i=1,m                                                      ma101600
      wx=a(i,jx)                                                        ma101610
      a(i,jx)=a(i,k)                                                    ma101620
  180 a(i,k)=wx                                                         ma101630
      interchange column norms.                                         ma101640
      wx=x(jx)                                                          ma101650
      x(jx)=x(k)                                                        ma101660
      x(k)=wx                                                           ma101670
      record column interchanges                                        ma101680
      iwx=iw(jx)                                                        ma101690
      iw(jx)=iw(k)                                                      ma101700
      iw(k)=iwx                                                         ma101710
      interchange column scale factors.                                 ma101720
      wx=w(n+jx)                                                        ma101730
      w(n+jx)=w(n+k)                                                    ma101740
      w(n+k)=wx                                                         ma101750
                                                                        ma101760
      calculate sigma, beta and transformation vector u(i) (see write   ma101770
      up of the method).                                                ma101780
                                                                        ma101790
  190 x(k)=fm02as(mx1-k,a(k,k),1,a(k,k),1)                              ma101800
      sigma=sqrt(x(k))*sign(1.,a(k,k))                                  ma101810
      u(k) stored in w(k) and u(i) in a(i,k)  i.gt.k                    ma101820
      w(k)=sigma+a(k,k)                                                 ma101830
      beta=1./(sigma*w(k))                                              ma101840
      pivot element.                                                    ma101850
      a(k,k)=-sigma                                                     ma101860
      k1=k+1                                                            ma101870
      if this is the last transformation of this step branch            ma101880
      if(k.eq.nx) go to 220                                             ma101890
      apply transformation.  h = scalar product (u,a(j))                ma101900
      do 210 j=k1,n                                                     ma101910
      h=fm02as(mx-k,a(k1,k),1,a(k1,j),1)                                ma101920
      h=beta*(h+w(k)*a(k,j))                                            ma101930
      do 200 i=k1,mx                                                    ma101940
  200 a(i,j)=a(i,j)-a(i,k)*h                                            ma101950
  210 a(k,j)=a(k,j)-w(k)*h                                              ma101960
      go to 150                                                         ma101970
  220 if(nx.eq.n) go to 300                                             ma101980
c     finish off transformation.                                        ma101990
      do 230 j=k1,n                                                     ma102000
  230 a(k,j)=(1.  -beta*w(k)*w(k))*a(k,j)                               ma102010
c                                                                       ma102020
c                                                                       ma102030
c     begin elimination of first m  variables in equations of condition ma102040
c                                                                       ma102050
c     form multipliers for elimination.                                 ma102060
  240 do 250 i=m11,m                                                    ma102070
  250 a(i,1)=a(i,1)/a(1,1)                                              ma102080
      if(m1.eq.1) go to 270                                             ma102090
      do 260 j=2,m1                                                     ma102100
      do 260 i=m11,m                                                    ma102110
      wx=a(i,j)-fm02as(j-1,a(i,1),1a,a(1,j),1)                          ma102120
  260 a(i,j)=wx/a(j,j)                                                  ma102130
  270 if(neqm1) go to 320                                               ma102140
c     carry out elimination.                                            ma102150
      do 280 j=m11,n                                                    ma102160
      do 280 i=m11,m                                                    ma102170
  280 a(i,j)=a(i,j)-fm02as(m1,a(i,1),1a,a(1,j),1)                       ma102180
c                                                                       ma102190
c                                                                       ma102200
c     begin transformation step q2 on equations of condition.           ma102210
c                                                                       ma102220
      k=k+1                                                             ma102230
  290 mx=m                                                              ma102240
      nx=n                                                              ma102250
      mx1=mx+1                                                          ma102260
      go to 130                                                         ma102270
c                                                                       ma102280
c                                                                       ma102290
c     start processing right hand side.                                 ma102300
c                                                                       ma102310
c     beta(k) stored in w(n+1)  i=1 to n                                ma102320
  300 do 310 j=1,n                                                      ma102330
  310 w(n+j)=-1.  /(w(j)*a(j,j))                                        ma102340
      if(res.and.neqm1.and..not.meqn) go to 240                         ma102350
  320 iswtch=.true.                                                     ma102360
c                                                                       ma102370
c     apply transformation q1 to first m1 right hand sides.             ma102380
      k=1                                                               ma102390
      nx=m1                                                             ma102400
      mx=m1                                                             ma102410
      if(m1-1)410,380,330                                               ma102420
  330 k1=k+1                                                            ma102430
      h=fm02as(mx-k,a(k1,k),1,b(k1),1)                                  ma102440
      h=w(n+k)*(h+w(k)*b(k))                                            ma102450
      do 340 i=k1,mx                                                    ma102460
  340 b(i)=b(i)-a(i,k)*h                                                ma102470
      b(k)=b(k)-w(k)*h                                                  ma102480
      if(iswtch) go to 350                                              ma102490
c     back transform for residuals in progress.                         ma102500
      k=k-1                                                             ma102510
      if(k.gt.m1) go to 330                                             ma102520
      go to 500                                                         ma102530
  350 k=k+1                                                             ma102540
  360 if(k-nx)330,370,420                                               ma102550
  370 if(k.lt.mx) go to 330                                             ma102560
```

```
c                                                                       ma101290
c                                                                       ma101300
c     begin transformation step q1 on constraint equations.             ma101310
c                                                                       ma101320
      nx=m1                                                             ma101330
      mx=m1                                                             ma101340
      mx1=mx+1                                                          ma101350
c                                                                       ma101360
c     calculate initial partial column norms and select largest.        ma101370
c     solution array used as tempory storage space for norms.           ma101380
  130 amx=0.                                                            ma101390
      do 140 j=k,n                                                      ma101400
      x(j)=fm02as(mx1-k,a(k,j),1,a(k,j),1)                              ma101410
      test= abs(x(j))*w(n+j)                                            ma101420
      if(test.le.amx) go to 140                                         ma101430
      jx=j                                                              ma101440
      amx=test                                                          ma101450
  140 continue                                                          ma101460
      go to 170                                                         ma101470
      calculate partial column norms using old values and select largestma101480
  150 k=k+1                                                             ma101490
      amx=0.                                                            ma101500
      do 160 j=k,n                                                      ma101510
      x(j)=x(j)-a(k-1,j)*a(k-1,j)                                       ma101520
      test= abs(x(j))*w(n+j)                                            ma101530
      if(test.le.amx) go to 160                                         ma101540
      jx=j                                                              ma101550
      amx=test                                                          ma101560
  160 continue                                                          ma101570
  170 if(jx.eq.k) go to 190                                             ma101580
      interchange columns so that col. with largest norm is used as pivoma101590
      do 180 i=1,m                                                      ma101600
      wx=a(i,jx)                                                        ma101610
      a(i,jx)=a(i,k)                                                    ma101620
  180 a(i,k)=wx                                                         ma101630
      interchange column norms.                                         ma101640
      wx=x(jx)                                                          ma101650
      x(jx)=x(k)                                                        ma101660
      x(k)=wx                                                           ma101670
      record column interchanges.                                       ma101680
      iwx=iw(jx)                                                        ma101690
      iw(jx)=iw(k)                                                      ma101700
      iw(k)=iwx                                                         ma101710
      interchange column scale factors.                                 ma101720
      wx=w(n+jx)                                                        ma101730
      w(n+jx)=w(n+k)                                                    ma101740
      w(n+k)=wx                                                         ma101750
c                                                                       ma101760
      calculate sigma, beta and transformation vector u(i) (see write   ma101770
      up of the method).                                                ma101780
c                                                                       ma101790
  190 x(k)=fm02as(mx1-k,a(k,k),1,a(k,k),1)                              ma101800
      sigma=sqrt(x(k))*sign(1.,a(k,k))                                  ma101810
      u(k) stored in w(k) and u(i) in a(i,k)  i.gt.k                    ma101820
      w(k)=sigma+a(k,k)                                                 ma101830
      beta=1./(sigma*w(k))                                              ma101840
      pivot element.                                                    ma101850
      a(k,k)=-sigma                                                     ma101860
      k1=k+1                                                            ma101870
      if this is the last transformation of this step branch            ma101880
      if(k.eq.nx) go to 220                                             ma101890
      apply transformation.  h = scalar product (u,a(j))                ma101900
      do 210 j=k1,n                                                     ma101910
      h=fm02as(mx-k,a(k1,k),1,a(k1,j),1)                                ma101920
      h=beta*(h+w(k)*a(k,j))                                            ma101930
      do 200 i=k1,mx                                                    ma101940
  200 a(i,j)=a(i,j)-a(i,k)*h                                            ma101950
  210 a(k,j)=a(k,j)-w(k)*h                                              ma101960
      go to 150                                                         ma101970
  220 if(nx.eq.n) go to 300                                             ma101980
c     finish off transformation.                                        ma101990
      do 230 j=k1,n                                                     ma102000
  230 a(k,j)=(1.  -beta*w(k)*w(k))*a(k,j)                               ma102010
c                                                                       ma102020
c                                                                       ma102030
c     begin elimination of first m1 variables in equations of condition ma102040
c                                                                       ma102050
c     form multipliers for elimination.                                 ma102060
  240 do 250 i=m11,m                                                    ma102070
  250 a(i,1)=a(i,1)/a(1,1)                                              ma102080
      if(m1.eq.1) go to 270                                             ma102090
      do 260 j=2,m1                                                     ma102100
      do 260 i=m11,m                                                    ma102110
      wx=a(i,j)-fm02as(j-1,a(i,1),ia,a(1,j  1)                          ma102120
  260 a(i,j)=wx/a(j,j)                                                  ma102130
  270 if(neqm1) go to 320                                               ma102140
c     carry out elimination.                                            ma102150
      do 280 j=m11,n                                                    ma102160
      do 280 i=m11,m                                                    ma102170
  280 a(i,j)=a(i,j)-fm02as(m1,a(i,1),ia,a(1,j),1)                       ma102180
c                                                                       ma102190
c                                                                       ma102200
c     begin transformation step q2 on equations of condition.           ma102210
c                                                                       ma102220
      k=k+1                                                             ma102230
  290 mx=m                                                              ma102240
      nx=n                                                              ma102250
      mx1=mx+1                                                          ma102260
      go to 130                                                         ma102270
c                                                                       ma102280
c                                                                       ma102290
c     start processing right hand side.                                 ma102300
c                                                                       ma102310
c     beta(k) stored in w(n+1)  i=1 to n                                ma102320
  300 do 310 j=1,n                                                      ma102330
  310 w(n+j)=-1.  /(w(j)*a(j,j))                                        ma102340
      if(res.and.neqm1.and..not.meqn) go to 240                         ma102350
  320 iswtch=.true.                                                     ma102360
c                                                                       ma102370
c     apply transformation q1 to first m1 right hand sides.             ma102380
      k=1                                                               ma102390
      nx=m1                                                             ma102400
      mx=m1                                                             ma102410
      if(m1-1)410,380,330                                               ma102420
  330 k1=k+1                                                            ma102430
      h=fm02as(mx-k,a(k1,k),1,b(k1),1)                                  ma102440
      h=w(n+k)*(h+w(k)*b(k))                                            ma102450
      do 340 i=k1,mx                                                    ma102460
  340 b(i)=b(i)-a(i,k)*h                                                ma102470
      b(k)=b(k)-w(k)*h                                                  ma102480
      if(iswtch) go to 350                                              ma102490
c     back transform for residuals in progress.                         ma102500
      k=k-1                                                             ma102510
      if(k.gt.m1) go to 330                                             ma102520
      go to 500                                                         ma102530
  350 k=k+1                                                             ma102540
  360 if(k-nx)330,370,420                                               ma102550
  370 if(k.lt.mx) go to 330                                             ma102560
```

```
  380 b(k)=b(k)*(1.-w(n+k)*w(k)*w(k))                                    ma102570
      k=k+1                                                              ma102580
      if(nx.eq.n) go to 420                                              ma102590
c                                                                        ma102600
c     elimination step for right hand sides.                             ma102610
  390 do 400 i=ml1,m                                                     ma102620
  400 b(i)=b(i)-fm02as(ml,a(1,1),ia,b(1),1)                              ma102630
  410 if(neqml) go to 430                                                ma102640
c                                                                        ma102650
c     begin applying transformation q2                                   ma102660
      nx=n                                                               ma102670
      mx=m                                                               ma102680
      go to 360                                                          ma102690
  420 if(res.and.neqml.and..not.meqn) go to 390                          ma102700
c                                                                        ma102710
c     back substitution to obtain solution which is stored temporarily   ma102720
c     in b(i)  i=1 to n.                                                 ma102730
  430 b(n)=b(n)/a(n,n)                                                   ma102740
      if(n.eq.1) go to 450                                               ma102750
      do 440 ii=1,nn                                                     ma102760
      i=n-ii                                                             ma102770
      wx=b(i)-fm02as(n-i,a(i,i+1),ia,b(i+1),1)                           ma102780
  440 b(i)=wx/a(i,i)                                                     ma102790
c                                                                        ma102800
c                                                                        ma102810
c       finish off and compute residuals if required                     ma102820
c                                                                        ma102830
c     put the solution in its correct order and store in x(j) j=1,n      ma102840
  450 do 460 i=1,n                                                       ma102850
      ix=iw(i)                                                           ma102860
      x(ix)=b(i)                                                         ma102870
  460 continue                                                           ma102880
      if(.not.res) go to 500                                             ma102890
c     residuals required                                                 ma102900
  470 do 480 i=1,n                                                       ma102910
  480 b(i)=0.                                                            ma102920
      if(meqn) go to 500                                                 ma102930
      do 490 i=ml1,m                                                     ma102940
  490 b(i)=-b(i)                                                         ma102950
      if(neqml) go to 500                                                ma102960
      iswtch=.false.                                                     ma102970
      mx=m                                                               ma102980
      k=n                                                                ma102990
      go to 330                                                          ma103000
  500 if(vflag) go to 550                                                ma103010
c                                                                        ma103020
c                                                                        ma103030
c     print solution and residuals.                                      ma103040
c                                                                        ma103050
      if(ipr.gt.0) return                                                ma103060
      write(6,510) name                                                  ma103070
  510 format('0',a8,1x,'solution')                                       ma103080
      call oa02a(x,n,10,6)                                               ma103090
      if(.not.res) go to 530                                             ma103100
      write(6,520) name                                                  ma103110
  520 format('0',a8,1x,'residuals')                                      ma103120
      call oa02a(b,m,10,6)                                               ma103130
      main return point to calling program.                              ma103140
  530 return                                                             ma103150
c                                                                        ma103160
c                                                                        ma103170
c       section for obtaining variance-covariance matrix                 ma103180
c                                                                        ma103190
  540 if(meqn) go to 700                                                 ma103200
c                                                                        ma103210
c       compute estimate of residual variance                            ma103220
      if(.not.res) go to 470                                             ma103230
  550 sumsq=fm02as(m,b(1),1,b(1),1)                                      ma103240
      var=sumsq/max0(1,m-n)                                              ma103250
      if(neqml) go to 710                                                ma103260
c                                                                        ma103270
c       let r be upper triangle of a then invert r                       ma103280
      do 560 ii=1,n                                                      ma103290
c       save interchanges                                                ma103300
      std(ii)=iw(ii)+.5                                                  ma103310
      i=n1-ii                                                            ma103320
      v(i,i)=1./a(i,i)                                                   ma103330
      if(i.eq.1) go to 570                                               ma103340
      ii=i-1                                                             ma103350
      do 560 jj=1,ii                                                     ma103360
      j=i-jj                                                             ma103370
      wk1=-fm02as(i-j,a(j,j+1),ia,v(j+1,i),1)                            ma103380
  560 v(j,i)=wk1/a(j,j)                                                  ma103390
c                                                                        ma103400
c         now form (r**-1)(k)trans(r**-1),  where (k) is                 ma103410
c         the identity matrix with first ml diagonals zero               ma103420
  570 do 580 i=1,n                                                       ma103430
      do 580 j=1,n                                                       ma103440
      k=max0(j,ml+1)                                                     ma103450
      v(i,j)=fm02as(n-k+1,v(i,k),iv,v(j,k),iv)                           ma103460
      v(i,j)=v(i,j)*var                                                  ma103470
  580 v(j,i)=v(i,j)                                                      ma103480
c                                                                        ma103490
c       re-order rows and columns                                        ma103500
      do 630 i=1,n                                                       ma103510
  590 if(i.eq.iw(i)) go to 620                                           ma103520
      k=iw(i)                                                            ma103530
      do 600 j=1,n                                                       ma103540
      wx=v(j,i)                                                          ma103550
      v(j,i)=v(j,k)                                                      ma103560
  600 v(j,k)=wx                                                          ma103570
      do 610 j=1,n                                                       ma103580
      wx=v(i,j)                                                          ma103590
      v(i,j)=v(k,j)                                                      ma103600
  610 v(k,j)=wx                                                          ma103610
      iwx=iw(i)                                                          ma103620
      iw(i)=iw(k)                                                        ma103630
      iw(k)=iwx                                                          ma103640
      go to 590                                                          ma103650
c       restore interchanges                                             ma103660
  620 iw(i)=std(i)                                                       ma103670
      std(i)=sqrt(v(i,i))                                                ma103680
  630 continue                                                           ma103690
c                                                                        ma103700
  640 if(ipr.gt.0) return                                                ma103710
      write(6,650) var                                                   ma103720
  650 format('0 mal4c: residual variance estimate',e14.6/                ma103730
     *       '0 mal4c: solution standard deviations')                    ma103740
      write(6,660) (std(j),j=1,n)                                        ma103750
  660 format(8x,e14.6)                                                   ma103760
      write(6,670)                                                       ma103770
  670 format('0 mal4c: variance-covariance matrix')                      ma103780
      do 680 i=1,n                                                       ma103790
  680 write(6,690) (v(i,j),j=1,n)                                        ma103800
  690 format(2x,9e14.6)                                                  ma103810
      return                                                             ma103820
c                                                                        ma103830
c       special case m = n                                               ma103840
```

```
  700 var=0.                                                          ma103850
        special case ml = n                                           ma103860
  710 do 720 i=1,n                                                    ma103870
      do 720 j=1,n                                                    ma103880
  720 v(i,j)=0.                                                       ma103890
      go to 640                                                       ma103900
                                                                      ma103910
        error conditions                                              ma103920
                                                                      ma103930
  730 write(6,740) name                                               ma103940
  740 format('0',a8,' error: argument restrictions violated')         ma103950
      mm=m-ml                                                         ma103960
      write(6,60) name,ml,mm,n                                        ma103970
      go to 530                                                       ma103980
      end                                                             ma103990
*_____________________________________________________________*
name oa02a(r)
      subroutine oa02a(b,m,nc,nsfa)                                   oa000050
      integer fmt,itrans                                              oa000060
      dimension b(1),fmt(9),itrans(24)                                oa000070
      data fmt/4h(3x,,1h4,2h(i,1h5,2h,d,2h24,1h.,2h16,2h))/           oa000080
      data itrans/1h1,1h2,1h3,1h4,1h5,1h6,1h7,1h8,1h9,                oa000090
     *           2h10,2h11,2h12,2h13,2h14,2h15,2h16,2h17,2h18,2h19,   oa000100
     *           2h20,2h21,2h22,2h23,2h24/                            oa000110
      nsf=max0(1,min0(nsfa,16))                                       oa000120
      nf=nsf+8                                                        oa000130
      fmt(6)=itrans(nf)                                               oa000140
      fmt(8)=itrans(nsf)                                              oa000150
      m10=m                                                           oa000160
      i10=2                                                           oa000170
   10 m10=m10/10                                                      oa000180
      i10=i10+1                                                       oa000190
      if(m10.ne.0)goto 10                                             oa000200
      k=126/(nf+i10)                                                  oa000210
      fmt(4)=itrans(i10)                                              oa000220
      fmt(2)=itrans(k)                                                oa000230
      nsfr=nsf                                                        oa000240
   20 im=(m-1)/nc+1                                                   oa000250
      ik=(im-1)/k+1                                                   oa000260
      it=m-(im-1)*nc                                                  oa000270
      itab=0                                                          oa000280
      i1=nc                                                           oa000290
      i3=nc*(k-1)                                                     oa000300
   30 if(ik-1) 90,40,50                                               oa000310
   40 i1=it                                                           oa000320
      i3=(im-1)*nc                                                    oa000330
   50 do 60 i=1,i1                                                    oa000340
      i2=itab+i                                                       oa000350
      i4=i2+i3                                                        oa000360
   60 write(6,fmt) (j,b(j),j=i2,i4,nc)                                oa000370
      if(im.eq.1.or.i1.eq.nc)goto 80                                  oa000380
      i31=i3-nc                                                       oa000390
      i11=i1+1                                                        oa000400
      do 70 i=i11,nc                                                  oa000410
      i2=itab+i                                                       oa000420
      i4=i2+i31                                                       oa000430
   70 write(6,fmt) (j,b(j),j=i2,i4,nc)                                oa000440
   80 itab=itab+nc*k                                                  oa000450
      ik=ik-1                                                         oa000460
      im=im-k                                                         oa000470
      write(6,999)                                                    oa000480
  999 format('0')                                                     oa000490
      goto 30                                                         oa000500
   90 return                                                          oa000510
      end
*_____________________________________________________________*
      function fm02as(n,a,ia,b,ib)
      real*8 fm02as
      real a(1),b(1)
      fm02as=0
      do 1 k=1,n
      fm02as=fm02as+a(1+(k-1)*ia)*b(1+(k-1)*ib)
    1 continue
      return
      end
```

```
      subroutine show(text1,ihardco)
*     ______________
*     triangulation of initial points and adding of dual points
*      flat case
*     'nost' - # points, 'r' - their coordinates
*     'ntr' - # triangles
*     next - pointers for neighbouring triangles
*     sites - pointers for points of the triangle
*     radius - radius of circle for glen triangle
*     center - its center
*     maxst - maximal number of points in the model
#include<fgl.h>
#include<fdevice.h>
      parameter(maxst=70000
     .         ,mmaxtr=2*maxst)
       common /points/ nost,ntr
       character*50 text1,filename
      real r(3,maxst)
      integer next(3,mmaxtr),sites(3,mmaxtr)
      real radius(mmaxtr),center(2,mmaxtr)
      real radv(mmaxtr),iord(mmaxtr)
*             pepectpoitx tpiahgulqci'
       common/lala/next,sites,r
       common /work1/ radius,center, radv,iord
      integer systeme

      common /lider/ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog

       common /win/ igid

       open(1,file='rescales',status='old',form='unformatted')
       open(2,file='tempor',status='unknown',form='unformatted')
       open(7,file='triang')
       open(4,file='leader')
      call glider

       do 10 ii=1,maxst-10
          read (1  ,end=1020,err=1010) r(1,ii),r(2,ii),r(3,ii)
   10     continue
c     print *,'===> dimension maxst too small    <---'
      stop
 1010 print *,'   ===>  error reading        <---'
      stop
 1020 nost=ii-1
 1030 continue
c     print *,'   ===>',nost,' points have been read      <---'

       call triala(r,next,sites,radius,center)
        call tryl(text1)
        call windo(r,next,sites,radius,center)
        call triala(r,next,sites,radius,center)

      print *,' triangulation of initial points is constructed'
      idr1=idra
      if(idra.le.0) then
*       print *, ' draw the triangulation ?'
*            call getans(idr1)
        endif
      if(idr1.ne.0) then
        call dsna(200,r)
        endif
      if(rdmax0.lt.0) then
c       print *,' radius of confidence ?'
c       read *,rdmax0
        endif
      rdmax=rdmax0**2
*     definition of radius of confidence
        if(rdmax.le.1e-50) then
            call detrad(rdmax,radius,radv,iord)
            rdmax0=sqrt(rdmax)
            endif
      call output(r,next,sites,radius,iord,ihardco)
      call plider(' triang ')
c     print *,'====>   end of job   <====='
      call winclo(igid)
      close(1)
      close(2)
      close(7)
      return
      end

      subroutine triala(r,next,sites,radius,center)
#include<fgl.h>
#include<fdevice.h>
      common /points/ nost,ntr
      common mccoi
      real backvec(3) /0.0,0.0,0.0/
      real r(3,1)
      integer next(3,1),sites(3,1)
      real radius(1),center(2,1)
      integer tgl
      logical ok
*
      nost00=nost
      call initt(r,next,sites,radius,center)


      tgl=1
      ndp=0
      do 20 np=2,nost00
          if(nost.ge.1000.and.mod(np,1000).eq.0) then
c             print *,'   ===>',np,' points have been included  <---'
              end if
          call search(tgl,np,ok,mk,r,next,sites)
          if(ok) then
              call bar(tgl,np,mk,r,next,sites,radius,center)
              call recon(tgl,r,next,sites,radius,center)
           else
              ndp=ndp+1
              end if
   20     continue
      return
      end


      subroutine dtris(kt,int,r,sites)
#include<fgl.h>
#include<fdevice.h>
        parameter(ncol=13)
      common /points/ nost,ntr
        common /mode/ exitmode,moda,pointmode,icount
        real legend(3,ncol)/
     &  0.0 ,0.0,0.0 ,
     &  0.0 ,0.0,0.8,
     &  0.0 ,0.0,1.0 ,
     &  0.0 ,0.6,0.6 ,
```

```
     &  0.0 ,0.8,0.0 ,
     &  0.0 ,1.0,0.0 ,
     &  0.2 ,0.9,0.0 ,
     &  1.0 ,1.0,0.0 ,
     &  1.0 ,0.8,0.0 ,
     &  1.0 ,0.6,0.0 ,
     &  1.0 ,0.4,0.0 ,
     &  1.0 ,0.2,0.0 ,
     &  1.0 ,0.0,0.0 /
        real r(3,1)
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      integer sites(3,1)
      real x(2),y(3),z(3)
      real backvec(3)/0.0,0.0,0.0/
      real bordovec(3)/0.3,0.0,0.3/
      mm(i)=mod(1,3)+1
        sca = (ncol-2)/(zmax-zmin)
      call c3f(backvec)
      call clear
      mhru = 1
      do 30 k=1,2
      do 10 nt=1,ntr
      zl=1
      do 19 i = 1,3
        if(r(3,sites(mm(i),nt)).lt.zmin-9)zl=0
19    continue
      if(zl.lt.1) then
      if(k.eq.1) then
       if(moda .eq.0) then
      call bgnclo
         else
         call bgnpol
         end if
         do 21 i=1,3
         x(1)=r(1,sites(mm(i),nt))
         x(2)=r(2,sites(mm(i),nt))
         if(x(1).lt.0.or.x(1).gt.1) goto 21
         if(x(2).lt.0.or.x(2).gt.1) goto 21
         if(moda .eq.0) then
         call c3f(bordovec)
         else
         call c3f(backvec)
         end if
        call  v2f(x)
1       continue
        if(moda .eq.0) then
      call endclo
         else
         call endpol
         end if

       end if
       goto 10
       end if


      if(moda .eq.0)then
      call bgnclo
      else
      call bgnpol
      end if
         do 20 i=1,3
          mcolx= (sca*(r(3,sites(mm(i),nt)-zmin)))+2.0
           x(1)=r(1,sites(mm(i),nt))
           x(2)=r(2,sites(mm(i),nt))
           if(x(1).lt.0.or.x(1).gt.1) goto 20
           if(x(2).lt.0.or.x(2).gt.1) goto 20
           call c3f(legend(1,mcolx))
         call  v2f(x)
20        continue
      if(moda .eq.0)then
        call endclo
      else
      call endpol
      end if
10    continue
30    continue
      return
      end


      subroutine chunks(r,mst)
        parameter(ncol=13)
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      common /points/ nost,ntr
        real legend(3,ncol)/
     &  0.0 ,0.0,0.0,
     &  0.0 ,0.0,0.8,
     &  0.0 ,0.0,1.0,
     &  0.0 ,0.6,0.6,
     &  0.0 ,0.8,0.0 ,
     &  0.0 ,1.0,0.0 ,
     &  0.2 ,0.9,0.0 ,
     &  1.0 ,1.0,0.0 ,
     &  1.0 ,0.8,0.0 ,
     &  1.0 ,0.6,0.0 ,
     &  1.0 ,0.4,0.0 ,
     &  1.0 ,0.2,0.0 ,
     &  1.0 ,0.0,0.0/
        real r(3,1)
        sca = ncol/(zmax-zmin)
        mcol = int((r(3,nt)-zmin)*sca)
        radius=.1/sqrt(1.*nost)
        do 10 nt=1,mst
        call c3f(legend(1,mcol))
        call circf(r(1,nt),r(2,nt),0.5*radius)
10       continue
      return
      end

      subroutine dsna(int,r)
      common /points/ nost,ntr
      real r(3,1)
*     call virs
      do 10 np=1,nost
         x=r(1,np)
         y=r(2,np)
         if(x.lt.0.or.x.gt.1) goto 10
         if(y.lt.0.or.y.gt.1) goto 10
      i1=x*255
      i2=255-y*255
10    continue
      return
      end
```

```
      subroutine windo(r,next,sites,radius,center)
include<fgl.h>
include<fdevice.h>
      common/popup_stuff/ menu,submenu

      parameter(ncol= 13,step = 1.0/ncol,Delta= 0.007)
      common /win/ igid
      real r(3,1)
     integer next(3,1),sites(3,1)
     real radius(1),center(2,1)
      character*1 string1*14,string2*25,string3*25
      common /points/ nost,ntr
       common mccol
      common /mode/ exitmode,moda,pointmode,icount
     common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      real backvec(3)/0.0,0.0,0.0/
      real yelvec(3)/1.0,1.0,0.0/
      real bluevec(3)/0.0,0.0,1.0/
       call prefpo(0,1270,0,1000)
       call foregr
      igid =  winope("triangulation",13)
      call winset(igid)
      call RGBmod
      call double
      call gconfi
      call c3f(backvec)
      call clear
      call swapbu
      mccol=-1
      exitmode = 0
      moda = 0
      pointmode = 1
      call qdevic(LEFTMO)
      call qdevic(MIDDLE)
      call qdevic(RIGHTM)
      call  tie(LEFTMO,MOUSEX,MOUSEY)
      call  tie(MIDDLE,MOUSEX,MOUSEY)

      menu = newpup()
      call addtop(menu,' DATA CHANGE %t|add point|copy point',50,0)
      call addtop(menu,' remove point |WIRED MODEL|MAP|EXIT ',50,0)
      call ortho(0.0,1.3,0.0,1.0,-1.0,1.0)
      do while (1)
      do while (qtest(0).eq.0)
      call Drawobj(r,next,sites,radius,center)
      end do
      call processinput(r,next,sites,radius,center)
      if(exitmode .eq.1) then
      goto 1045
      end if
      end do
1045  continue
      return
      end



      subroutine Drawobj(r,next,sites,radius,center)
      parameter(ncol= 13,step = 1.0/ncol,Delta= 0.007)
      real r(3,1)
      integer sites(3,1)
      character*1 string1*14,string2*25,string3*25
      common /points/ nost,ntr
       common mccol
      common /mode/ exitmode,moda,pointmode,icount
     common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      real backvec(3)/0.0,0.0,0.0/
      real yelvec(3)/1.0,1.0,0.0/
      real bluevec(3)/0.0,0.0,1.0/
      call dtris(0,int,r,sites)
      call palette
      write(string1,*)
   12 format(f8.4)
      call cmov2(1.04,0.50)
      call charst(string1,4)
      write (string2,*) 'Hit the right button'
      write (string3,*) ' to get the menu '
      call c3f(bluevec)
      call rectf(1.01,0.95,1.25,0.99)
      call c3f(yelvec)
       call  cmov2(1.02,0.96)
      call charst(string3,23)
       call  cmov2(1.02,0.98)
      call charst(string2,23)

      call swapbu
      return
      end

      subroutine processinput(r,next,sites,radius,center)
#include <fgl.h>
#include <fdevice.h>
      integer*2 val,mx,my,mcx,mcy
      integer dev
      integer do_job ,ch_col
      integer menuval
      common /mode/ exitmode,moda,pointmode,icount
      common/popup_stuff/ menu,submenu
      character*1  string2*30,string3*30

      real r(3,1)
     integer next(3,1),sites(3,1)
     real radius(1),center(2,1)

      do_job = 0
      ch_col = 0
      dev = qtest()
   12 format(f8.4)

      do  1000  while (dev .ne.0)
      dev = qread(val)
      if(dev .eq. RIGHTM) then
      menuval = dopup(menu)
       if (menuval .eq. 6) then
      exitmode = 1
      goto 999
      end if
       if (menuval .eq. 5) then
      moda = 1
      call Drawobj(r,next,sites,radius,center)
      end if
       if (menuval .eq. 4) then
      moda = 0
```

```
        call Drawobj(r,next,sites,radius,center)
        end if

* adding
        if(menuval .eq. 1) then
        pointmode = 1
        end if
* copying
        if(menuval .eq. 2)then
        pointmode = 2
        icount = 0
        end if
* removing
        if(menuval .eq. 3)then
        pointmode = 3
        end if
       else if(dev .eq. MIDDLE) then
       if (val .eq. 1) then
       l= qread(mcx)
       k= qread(mcy)
       call change_col(mcx,mcy,r,next,sites,radius,center)
       do_job = 0
       end if
       else if(dev .eq. LEFTMO) then
       if (val .eq. 1) then
       m = qread(mx)
       n = qread(my)
       do 1001  while(qtest() .eq. 0 )

        call palette

         call make_change(mx,my,r,next,sites,radius,center)

.001    end do
       end if
       end if




'000    end do

 999    continue
       return
       end


      subroutine palette
      parameter(ncol = 13,xl=1.17,yl =  0.07,six =  0.1)
       common mccol
    common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
   &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
   &              ,ipik,ipog
      real legend(3,ncol)/
   &  0.0 ,0.0,0.0 ,
   &  0.0 ,0.0,0.8,
   &  0.0 ,0.0,1.0 ,
   &  0.0 ,0.6,0.6 ,
   &  0.0 ,0.8,0.0 ,
   &  0.0 ,1.0,0.0 ,
   &  0.2 ,0.9,0.0 ,
   &  1.0 ,1.0,0.0 ,
   &  1.0 ,0.8,0.0 ,
   &  1.0 ,0.6,0.0 ,
   &  1.0 ,0.4,0.0 ,
   &  1.0 ,0.2,0.0 ,
   &  1.0 ,0.0,0.0 /

       real blackvec(3) /0.0,0.0,0.0/
       real redvec(3) /1.0,0.0,0.0/
       real backvec(3)/0.8,0.8,0.8/

      integer  i,j,k
      character*1  string*27,string1*4,string2*12,string3*12,str4*12
      string = ' zcoord'
      call c3f(backvec)
      call rectf(1.0,0.0,1.3,1.0)
      siy = (1.0 - 2*yl )/ncol
      k=0
      do 1 i=1,13
      k=k+1
      call c3f(legend(1,k))
      call rectf(xl,yl+(i-1)*siy,xl+six,yl+i* siy)
    1 continue
      call c3f (blackvec)
      call rect(xl ,yl,xl+six ,1.0-yl)
      call circ(1.08,0.5,0.05)
      call cmov2(1.04,0.43)
      call charst(string,6)
c     call c3f(legend(1,mccol))
c     call rectf(1.02,0.5,1.12,0.6)
      write(string2,13)zmin
      write(string3,13)zmax
      write(str4,*)'  Hole'
      call cmov2(xl-0.1,yl)
      call charst(str4,17)
      call cmov2(xl-0.1,2*yl)
      call charst(string2,17)
      call cmov2(xl-0.1,1-yl)
      call charst(string3,17)
   13 format(f8.4)
      return
      end



      subroutine change_col(xcol,ycol,r,next,sites,radius,center)
#include<fgl.h>
#include<fdevice.h>
      parameter(ncol = 13,xl=1.15,yl =  0.05,six =  0.1)
       common mccol
      common /points/ nost,ntr
    common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
   &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
   &              ,ipik,ipog
      character*1 string1*12,string2*12
      character*1  string4*12,string3*12
      real r(3,1)
     integer next(3,1),sites(3,1)
     real radius(1),center(2,1)
     integer tgl
     logical ok
       integer*2  xcol,ycol
```

```
      real xc,yc
      integer  i,j
      real legend(3,ncol)/
     &  0.0 ,0.0,0.0 ,
     &  0.0 ,0.0,0.8,
     &  0.0 ,0.0,1.0 ,
     &  0.0 ,0.6,0.6 ,
     &  0.0 ,0.8,0.0 ,
     &  0.0 ,1.0,0.0 ,
     &  0.2 ,0.9,0.0 ,
     &  1.0 ,1.0,0.0 ,
     &  1.0 ,0.8,0.0 ,
     &  1.0 ,0.6,0.0 ,
     &  1.0 ,0.4,0.0 ,
     &  1.0 ,0.2,0.0 ,
     &  1.0 ,0.0,0.0 /

      real backvec(3)/0.8,0.8,0.8/
      real blackvec(3)/0.0,0.0,0.0/

      mcold= mccol
      zdel = (zmax-zmin)/(ncol-2)

      siy = (1.0 - 2*yl )/ncol
      xc = (1.3/1270) * xcol
      yc = (1.0/1000) * ycol
      if(xc .lt. xl .or. xc.gt. (xl+six)  ) then
      mccol = -1
      else
      mccol = ((yc -yl)/siy)+1
      end if
      if(mccol .lt. 0 .or. mccol .gt. 13) then
      mccol = -1
      end if
      if(mccol.gt.1) then
      write(string1,12)zmin + (mccol)*zdel
      else
      if(mccol .gt.0)then
      write(string1,*)'  Hole'
      else
      write(string1,*)
      end if
      end if
      if( mcold .gt. 1) then
      write(string2,12)zmin + zdel*(mcold)
      else
      if( mcold .gt. 0) then
      write(string2,*)'  Hole'
      else
      write(string2,*)
      end if
      end if
12    format(f8.4)
      call frontb(.TRUE.)
      call c3f(backvec)
      call cmov2(1.04,0.5)
      call charst(string2,12)
      call c3f(blackvec)
      call cmov2(1.04,0.50)
      call charst(string1,12)

      call frontb(.FALSE.)
      return
      end



      subroutine make_change(x,y,r,next,sites,radius,center)
#include<fgl.h>
#include<fdevice.h>
      parameter(ncol = 13,Delta = 0.007)
       common mccol
      common /points/ nost,ntr
      common /mode/ exitmode,moda,pointmode, icount
     common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      integer*2  x,y
      character*1 string1*12,string2*30,string3*30,string4*30
      real r(3,1)
     integer next(3,1),sites(3,1)
     real radius(1),center(2,1)
     integer tgl
     logical ok

      real x3,y3
      real legend(3,ncol)/
     &  0.0 ,0.0,0.0 ,
     &  0.0 ,0.0,0.8,
     &  0.0 ,0.0,1.0 ,
     &  0.0 ,0.6,0.6 ,
     &  0.0 ,0.8,0.0 ,
     &  0.0 ,1.0,0.0 ,
     &  0.2 ,0.9,0.0 ,
     &  1.0 ,1.0,0.0 ,
     &  1.0 ,0.8,0.0 ,
     &  1.0 ,0.6,0.0 ,
     &  1.0 ,0.4,0.0 ,
     &  1.0 ,0.2,0.0 ,
     &  1.0 ,0.0,0.0 /

      real bluevec (3)/0.0,0.0,1.0/
      real yelvec  (3)/1.0,1.0,0.0/

      tgl = 1

      x3=(1.3/1270)*x
      y3= (1.0/1000)*y
* adding
      if(pointmode .eq. 1) then
      z = 0
      call add_point(x3,y3,z ,r,next,sites,radius,center)
      end if

* copy

      if(pointmode .eq.2) then
      icount = icount+1
      if(icount .eq. 1) then
      call copy_point(x3,y3,z,r,next,sites,radius,center)
      else
      call add_point(x3,y3,z,r,next,sites,radius,center)
      end if
      end if
```

```
* removing
      if(pointmode .eq.3) then
      call remove_point(x3,y3,r,next,sites,radius,center)
      end if

      delz =( zmax - zmin )/(ncol-2)
      call dtris(0,int,r,sites)
      call palette
      if(pointmode .eq.1) then
      if(mccol .gt.1)then
      write(string1,12) zmin+mccol*delz
      else
      if(mccol .eq.1)then
      write(string1,*) '  Hole'
       else
      write(string1,*)
      end if
      end if
      end if
      if(pointmode.eq.2)then
      write(string4,*) 'The point is chosen'
      write(string1,12)  z
      end if

 12    format(f8.4)
       call cmov2(1.04,0.50)
      call charst(string1,12)
      call c3f(bluevec)
      call rectf(1.01,0.95,1.25,0.99)
      call c3f(yelvec)
      write (string2,*) 'Hit right button'
      write (string3,*) 'to get menu'
       call  cmov2(1.02,0.96)
      call charst(string3,23)
       call  cmov2(1.02,0.98)
      call charst(string2,23)
      if(pointmode .eq.2) then
      call cmov2(1.001,0.57)
      call c3f(blackvec)
      call charst(string4,23)
      end if


      call swapbu
      return
      end

      subroutine vector(r,n1,n2,kt,l)
      parameter(ncol=13)
    common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &              madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &             ,ipik,ipog
      real legend(3,ncol)/
     & 0.0,0.0,0.0,
     & 0.0 ,0.0,0.8,
     & 0.0 ,0.0,1.0,
     & 0.0 ,0.6,0.6,
     & 0.0 ,0.8,0.0 ,
     & 0.0 ,1.0,0.0 ,
     & 0.2 ,0.9,0.0 ,
     & 1.0 ,1.0,0.0 ,
     & 1.0 ,0.8,0.0 ,
     & 1.0 ,0.6,0.0 ,
     & 1.0 ,0.4,0.0 ,
     & 1.0 ,0.2,0.0 ,
     & 1.0 ,0.0,0.0 /

      real link(3)/0.9,0.9,0.9/
      real backvec(3)/0.0,0.0,0.0/
      real r(3,1)
      x1=r(1,n1)
      x2=r(2,n1)
      x3=r(3,n1)
      y1=r(1,n2)
      y2=r(2,n2)
      y3=r(3,n2)
      sca = ncol/(zmax-zmin)
      mcolx= (x3-zmin)*sca
      mcoly= (y3-zmin)*sca
      if (kt.eq.0)then
       call c3f(link)
       else
       call c3f(backvec)
       end if
       call move(x1,x2,0.0)
       call draw(y1,y2,0.0)
c      call c3f(legend(1,mcolx))
c      call circf(x1,x2,.005)
c      call c3f(legend(1,mcoly))
c      call circf(y1,y2,.005)
       return
       end



      subroutine remove_point(x3,y3,r,next sites,radius,center)
#include<fgl.h>
#include<fdevice.h>
      parameter(ncol = 13,Delta = 0.007)
       common mccol
      common /points/ nost,ntr
      common /mode/ exitmode,moda,pointmode ,icount
    common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &              madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &             ,ipik,ipog
      real x3,y3
      real r(3,1),point(2)
     integer next(3,1),sites(3,1)
     real radius(1),center(2,1)
     integer tgl
     logical ok

       real blackvec  (3)/0.0,0.0,0.0/

       tgl = 1

       r(1,nost+1) = x3
       r(2,nost+1) = y3
       r(3,nost+1) = zmin - 10
       rad = Delta**2

         call search(tgl,nost+1,ok,mk,r,next,sites)
         call distance(r,sites,nost+1,tgl,rad,m)
         if(m .lt. 0) return

         call frontb(.TRUE.)
         call c3f(blackvec)
```

```
          call circf(r(1,sites(m,tgl)),r(2,sites(m,tgl)),Delta)
          call frontb(.FALSE.)
          n = sites(m,tgl)
* remove the point from the r
          do 10 i = n,nost -1
          do 20 k = 1,3
   20     r(k,i) = r(k,i+1)
   10     continue
          nost=nost-1


        call triala(r,next,sites,radius,center)




          return
          end



        subroutine add_point(x3,y3,z,r,next,sites,radius,center)
#include<fgl.h>
#include<fdevice.h>
        parameter(ncol = 13,Delta = 0.007)
         common mccol
        common /points/ nost,ntr
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
        real  x3,y3
        real r(3,1)
      integer next(3,1),sites(3,1)
      real radius(1),center(2,1)
      integer tgl
      logical ok

        real legend(3,ncol)/
     &  0.0 ,0.0,0.0 ,
     &  0.0 ,0.0,0.8,
     &  0.0 ,0.0,1.0 ,
     &  0.0 ,0.6,0.6 ,
     &  0.0 ,0.8,0.0 ,
     &  0.0 ,1.0,0.0 ,
     &  0.2 ,0.9,0.0 ,
     &  1.0 ,1.0,0.0 ,
     &  1.0 ,0.8,0.0 ,
     &  1.0 ,0.6,0.0 ,
     &  1.0 ,0.4,0.0 ,
     &  1.0 ,0.2,0.0 ,
     &  1.0 ,0.0,0.0 /


        tgl = 1

        call frontb(.TRUE.)
        call c3f(legend(1,mccol))
        call circf(x3,y3,Delta)
        call frontb(.FALSE.)
        if(mccol.gt.0) then
        if(mccol.eq.1) then
        nost = nost+1
          r(1,nost) = x3
          r(2,nost) = y3
          r(3,nost) = zmin-10
          else
          nost = nost +1
          delz =( zmax - zmin )/(ncol-2)
          r(1,nost) = x3
          r(2,nost) = y3
          if(pointmode .eq.2) then
          r(3,nost) = z
          else
          r(3,nost) = zmin + float(mccol-2) * delz
          end if
          end if
            call search(tgl,nost,ok,mk,r,next,sites)
            if(ok) then
                call bar(tgl,nost,mk,r,next,sites,radius,center)
                call recon(tgl,r,next,sites,radius,center)
            end if
          end if


           return
           end

        subroutine copy_point(x3,y3,z,r,next,sites,radius,center)
#include<fgl.h>
#include<fdevice.h>
        parameter(ncol = 13,Delta = 0.007)
         common mccol
        common /points/ nost,ntr
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
        real  x3,y3
        real r(3,1)
      integer next(3,1),sites(3,1)
      real radius(1),center(2,1)
      integer tgl
      logical ok
      character*40 string

        real legend(3,ncol)/
     &  0.0 ,0.0,0.0 ,
     &  0.0 ,0.0,0.8,
     &  0.0 ,0.0,1.0 ,
     &  0.0 ,0.6,0.6 ,
     &  0.0 ,0.8,0.0 ,
     &  0.0 ,1.0,0.0 ,
     &  0.2 ,0.9,0.0 ,
     &  1.0 ,1.0,0.0 ,
     &  1.0 ,0.8,0.0 ,
     &  1.0 ,0.6,0.0 ,
     &  1.0 ,0.4,0.0 ,
     &  1.0 ,0.2,0.0 ,
     &  1.0 ,0.0,0.0 /


        tgl = 1
        sca = (ncol-2)/(zmax-zmin)

        r(1,nost+1) = x3
        r(2,nost+1) = y3
        r(3,nost+1) = 0
```

```
      rad = Delta**2

        call search(tgl,nost+1,ok,mk,r,next,sites)
        call distance(r,sites,nost+1,tgl,rad,m)
        if(m .lt. 0) return

        n = sites(m,tgl)
      mccol= (sca*(r(3,sites(m,tgl)-zmin)))+2.0
        call frontb(.TRUE.)
        call c3f(legend(1,mccol))
        call circf(r(1,sites(m,tgl)),r(2,sites(m,tgl)),Delta)
        call frontb(.FALSE.)
        z = r(3,sites(m,tgl))


      return
      end
```

```
      subroutine start(filename,norm,w,ipeak)
*
*     ordering along the snake
      parameter(maxst=30000)
      parameter (maxpoi=35000,maxwor=10000)
c     parameter (nmin=1)
      parameter (nmin=16)
*     'nst' - # to-ek, 'r' - ix koopdihaty
*     maxpoi - number of points in the initial file
*     maxwor - dimension of working array
      parameter(maxlv=20)

      real*4 x(maxpoi),y(maxpoi)
      character*30   filename
      integer nnor(maxpoi)
      integer work(maxpoi+maxwor*3)
      integer iord(maxst*6+maxlv*6)
      real*4 r(2,maxst)
        real*4 f(nmin,maxst),ff(nmin),num(maxst)
        real*4 amem(maxpoi*3+maxwor*3)
        common/work1/ amem
        common/lala/ num,nnor
      equivalence (amem(1),x(1))
      equivalence (amem(maxpoi+1),y(1))
      equivalence (amem(maxpoi*2+1),work(1))
      equivalence (amem(1),iord(1))
      equivalence (amem(maxst*6+maxlv*6+1),r(1,1))
      equivalence (amem(maxst*8+maxlv*6+1),f(1,1))

      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,xva
     &              ,ipik,ipog
1     format(40a)
      open(2,file='rescales',status = 'unknown',form ='unformatted')
      open(4,file='leader')
      call glider
       ipik = ipeak
       if(ipeak .eq. 1)ipog = 1
      if(ipog.eq.0) then
         nmil=nmi0
      else
         nmil=nmi0*2
         endif
      if(nmil.gt.nmin) then
         print *,'===> dimension nmin is too small <==='
         print *,'===>     put     nmin = ',nmil,' <===='
         stop 18
         endif

      ii=int(log(real(maxst))/log(3.0))+1
      if (ii   .gt.   maxlv)  then
         print *,'===> dimension maxlv is too small <===='
         print *,'===>     put     maxlv = ',ii  ,' <===='
          stop
          endif

      nst=0
      if(ifo.eq.0) then
      open(1,file = filename,status= 'old',form = 'unformatted')
         do 10 i1=1,maxpoi+1
            read (1 , end=1020,err=1010) x(i1),y(i1)
            nst=nst+1
10          continue
      else
         open(1,file = filename)
            do 11 i1=1 maxpoi+1
               read (1,*,end=1020,err=1010) x(i1),y(i1)
               nst=nst+1
11             continue
            endif

            print *,'===> dimension maxpoi is too small <===='
            print *,'===> put maxpoi equal to amount of <===='
            print *,'===> points in input file          <===='
         stop 12
1010  print *,'    ===>  error reading        <===='
      stop 11
1020  continue
13    format(' ',i6,' points have been read')
      if(xmax**2+xmin**2.le.1e-50) then
         xmax=-1.e50
         xmin= 1.e50
         do 786 i1=1,nst
         xmax=max(xmax,x(i1))
         xmin=min(xmin,x(i1))
786      continue
         endif
      if(ymax**2+ymin**2.le.1e-50) then
         ymax=-1.e50
         ymin= 1.e50
         do 787 i1=1,nst
         ymax=max(ymax,y(i1))
         ymin=min(ymin,y(i1))
787      continue
         endif
      compr1=1./(xmax-xmin)
      compr2=1./(ymax-ymin)
      compr=min(compr1,compr2)
      sx=(1.-(xmax-xmin)*compr)/2
      sy=(1.-(ymax-ymin)*compr)/2

      call agrid(x,y,nst,nnor,nst1,work,maxpoi,maxwor)
      if(nst1.gt.maxst) then
         print *,'===> dimension maxst is too small <===='
         print *,'===>     put     maxst    = ',nst1,' <===='
         print *,'===> or decrease the size of grid <===='
         stop 6
         endif
         print *, 'line # 111 nst1 = ', nst1
      do 630 i=1,nst1
630      num(i)=0
      rewind 1
      do 747 i=1,nst
         if(ifo.eq.0) read(1 ) xx,yy,(ff(ij),ij=1,nmil)
         if(ifo.ne.0) read(1,*) xx,yy,(ff(ij),ij=1,nmil)
         kk=nnor(i)
         if(kk.lt.1) goto 747
         r(1,kk)=(r(1,kk)*num(kk)+xx)/(num(kk)+1.)
         r(2,kk)=(r(2,kk)*num(kk)+yy)/(num(kk)+1.)
         do 745 i1=1,nmil
            f(i1,kk)=(f(i1,kk)*num(kk)+ff(i1))/(num(kk)+1.)
745         continue
         num(kk)=num(kk)+1
747      continue
      nst=nst1
       print *, 'nst = ', nst
      do 788 i1=1,nst
      r(1,i1)=(r(1,i1)-xmin)*compr+sx
```

```
      r(2,i1)=(r(2,i1)-ymin)*compr+sy
      if(isu.ne.0) then
         do 783 i77=1,nmi1
         f(i77,i1)=f(i77,i1)*compr
 783     continue
         endif
 788  continue
         zmax=-1.e50
         zmin= 1.e50
         do 1786 i1=1,nst
         do 1786 i77=1,nmi1
         zmax=max(zmax,f(i77,i1))
         zmin=min(zmin,f(i77,i1))
1786      continue
   14   format(a,f8.3,a,f8.3)


         cola = (zmax-zmin)/log(w+1)
         do 1787 i1=1,nst
         do 1787 i77=1,nmi1
         f(i77,i1)=f(i77,i1)-zmin
         if(w .ge.1)then
         f(i77,i1) =1+w* f(i77,i1)/(zmax - zmin)
         f(i77,i1) = log(f(i77,i1))*cola
         zmax1 = log(1+w*zmax/(zmax-zmin))*cola
         zmin1 = log(1*w*zmin/(zmax-zmin))*cola
         zmax = zmax1
         zmin = zmin1
         end if
         if(norm .eq.1 ) f(i77,i1) = f(i77,i1)/(zmax-zmin)
1787      continue
         zmax = zmax -zmin
         zmin = 0




      call snake(r,iord,nst,maxlv,maxst)

      do 30 i=1,nst
      write (2) (r(j,iord(i)),j=1,2),(f(i,iord(i)),i=1,nmi1)
ccc     print *, (r(j,iord(i)),j=1,2),(f(i,iord(i)),i=1,nmi1)
   30 continue

      call plider(' snake  ')
      close(1)
      close(2)
      close(4)
      if (ind .eq.1) then
      endif
      return
      end

      subroutine agrid(x,y,nst,nnor,nst1,w,maxpoi,maxwor)
      integer w(1),nnor(1)
      real x(1),y(1)
      m1=1
      m2=m1+maxpoi
      m3=m2+maxwor
      m4=m3+maxwor
      call agri1(x,y,nst,nnor,nst1,w(m1),w(m2),w(m3),w(m4),maxwor)
      return
      end

      subroutine agri1(x,y,nst,nnor,nst1,ior1,ior2,npo,y1,maxwor)
      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      real*4 x(1),y(1)
      integer ior1(1),nnor(1),npo(1),ior2(1)
      real*4 y1(1)

      do 737 ki=1,nst
 737  nnor(ki)=0
       print *, 'grid = ', grid
      dd=max((xmax-xmin),(ymax-ymin))/grid
          call kb07a(x,nst,ior1)
          do 773 ip5=1,nst
          i1=ip5
          if(x(ip5).ge.xmin) goto 774
 773      continue
 774      continue
          stx=xmin
          kk=0
          icheck=0
 5           if(stx.gt.xmax) goto 100
          stx=stx+dd
          iy=0
          if(icheck.gt.0)  goto 100
          do 93 i=i1,nst
               i2=i
               ii=ior1(i)
              if(x(i).gt.stx) goto 35
              iy=iy+1
              npo(iy)=ii
              y1(iy)=y(ii)
 93       continue
            icheck=1
 35       continue
          if(iy.gt.maxwor) then
             print *,'===> dimension maxvor is too small <===='
             print *,'===> increase  maxwor              <===='
             print *,'===> or decrease the size of grid  <===='
            stop
            endif
          if(iy.eq.0) goto 5
            i1=i2
          call kb07a(y1,iy,ior2)
          sty=max(ymin,y1(1))+dd
          kk=kk+1
          do 43 i=1,iy
               if(y1(i).lt.ymin) goto 43
               if(y1(i).gt.sty) then
                  if(sty.gt.ymax) goto 5
782             sty=sty+dd
               if(y1(i).gt.sty) goto 782
               kk=kk+1
               endif
               nnor(npo(ior2(i)))=kk
 43         continue
           goto 5
 100  continue
      nst1=kk
      return
      end
```

```
      subroutine snake(r,w,nst,maxlv,maxst)
      integer w(1)
      real r(2,maxst)
      m1=1
      m2=m1+maxst*2
      m3=m2+maxst*2
      m4=m3+maxlv*4
      m5=m4+maxlv*2
      call snak1(r,w(m1),nst,w(m2),w(m3),w(m4),maxlv,maxst,w(m5))
      do 10 i=1,nst
      w(i)=w(i+maxst)
10    continue
      return
      end

      subroutine snak1(r,iord,nst,mord,corner,nn,maxlv,maxst,order)
      integer corner(4,maxlv),nn(2,maxlv),iord(maxst,2),mord(maxst,2)
      real r(2,maxst)
      real order(maxst*2)
c-----------------------------------------------------------------------$
      do 3 id=1,2
          do 6 i =1,nst
          order(i)=r(id,i)
          call kb07a(order,nst,iord(1,id ))
          do 33 mm=1,nst
              mord (iord(mm,id),id )=mm
33        continue
      call uprspl(nst,corner,nn,iord,mord,maxst,maxlv,r,order)
c     call snasna(nst,corner,nn,iord,mord,maxst,maxlv)
      return
      end

      subroutine uprspl(nst,corner,nn,iord,mord,maxst,maxlv,x,ibf)
       include(param)
      integer  corner(4,maxlv),nn(2,maxlv),iord(maxst,2),mord(maxst,2)
      integer ibf(maxst*2)
      real x(,)

**    begin
      lv=1
      nn(1,lv)=1
      nn(2,lv)=nst
      corner(1,lv)=1
      corner(2,lv)=1
1     continue
      if(nn(2,lv)-nn(1,lv).ge.1) then
        lv=lv+1
        call split (lv,corner,nn,iord,mord,maxst,maxlv,x,ibf)
      else
        n1=nn(2,lv)+1
        n2=min(2*nn(2,lv)+1-nn(1,lv),nn(2,lv-1))
        print *, 'from bor ',n1,n2
          if(n1.gt.n2) then
          lv=lv-1
          if(lv.eq.1) return
          goto 1
          else
          nn(1,lv)=n1
          nn(2,lv)=n2
          corner(1,lv)=corner(1,lv)+1
           endif
      endif
      goto 88
      end

***   end subroutine
*     ------------------------------------------------------------------$
      subroutine split (lv,corner,nn,iord,mord,maxst,maxlv,x,ibf)
*     include(param)
      logical test
      integer iadd(3),ibf(maxst,2)
     .,corner(4,maxlv),nn(2,maxlv),iord(maxst,2),mord(maxst,2)
      real x(2,maxst)
***   begin
      n1=nn(1,lv-1)
      n2=nn(2,lv-1)
c     print *,'  lv  n1 n2  '
*     print *,   lv, n1,n2
      if(n1.eq.n2) return
      k=corner(1+corner(1,lv-1),lv-1)
*  tip delehiq
      dx=x(1,iord(n2,1))-x(1,iord(n1,1))
      dy=x(2,iord(n2,2))-x(2,iord(n1,2))
      if(dx.gt.dy) then
      ityp=1
      else
      ityp=2
      endif
      test=ityp.eq.1.and.(k.eq.3.or.k.eq.4)
      test=test.or.ityp.eq.2.and.(k.eq.2.or.k.eq.3)
*   pointers
      corner(1,lv)=1
      corner(2,lv)=k
      corner(4,lv)=k
      if(ityp.eq.1  ) then
      corner(3,lv)=2*mod(k,2)-1+k
      else
      corner(3,lv)=5-k
      endif
*     split ------------------------------------------------------------$
      len =(n2-n1)/3+1
      nn(1,lv)=n1
      nn(2,lv)=len+n1-1
      m1=n1
      do 1 i=1,3
      iadd(i)=m1
      if(m1.gt.n2) goto 1
      m2=min(n2,m1+len-1)
      if(test) then
* reverse
      l1=max(n2-i*len+1,n1)
      else
      l1=m1
      endif
      do 2 j=m1,m2
      jj=l1+j-m1
      l1=iord(jj,ityp)
      ibf(j,ityp)=l1
      mord(l1,ityp)=j
2     continue
      m1=m2+1
1     continue
*     ------------------------------------------------------------------$
      jtyp=3-ityp
      do 3  j=n1,n2
      l1=iord(j,jtyp)
      jj=mord(l1,ityp)
      kk=(jj-n1)/len+1
      ibf (iadd(kk),jtyp)=l1
```

```
      mord(ll,jtyp)=iadd(kk)
      iadd(kk)=iadd(kk)+1
    4 continue
      do 33 j=n1,n2
      do 33 i= 1, 2
      iord(j,i   )=ibf(j,i)
   3  continue

      return
   ** end subroutine
      end
_____________________________________________________$
      subroutine snasna(n,corner,nn,iord,mord,maxst,maxlv)
      include(param)
      integer  corner(4,maxlv),nn(2,maxlv),iord(maxst,2),mord(maxst,2)
      logical direct
   ** begin snasna

      m=log(float(n))/log(2.)+1
      m=2**m
      mm=m
      iord(1,2)=iord(n,1)
      iord(2,2)=iord(m/2,1)
      j=2
      direct=.true.
      istep=m/4
      continue
      istep2=istep*2
      if(direct) then
      do 10  i=istep,n-1,istep2
      if(mod(i,m).eq.0) print *,'error'
      j=j+1
      iord(j,2)=iord(i,1)
   10 continue
      direct=.false.
         else
         kk=(mm+istep-n+.1)/istep2
         print *,'kk=',kk
         nearn=mm-kk*istep2-istep
      do 20  i=nearn,istep,(-istep2)
      if(mod(i,m).eq.0) print *,'error2','i=',i,m
      j=j+1
      iord(j,2)=iord(i,1)
   20 continue
      direct=.true.
         endif
      m=istep
      istep=istep/2
      if(istep.ge.1) goto 1
      open(7,file='tesn')
      do 39  i= 1 , nst
      write(7) i,iord(i,2)
    9  continue
      close(7)
      return
    * end subroutine snasna
      end
```

```
      subroutine trian(ihardco)
*
*     ____________
*     triangulation of initial points and adding of dual points
*     flat case
*     'nst' - # points, 'r' - their coordinates
*     'ntr' - # triangles
*     next - pointers for neighbouring triangles
*     sites - pointers for points of the triangle
*     radius - radius of circle for glen triangle
*     center - its center
*     maxst - maximal number of points in the model
      parameter(maxst=70000
     .         ,mmaxtr=2*maxst)
        common /points/ nst,ntr
      real r(3,maxst)
      integer next(3,mmaxtr),sites(3,mmaxtr)
      common/lala/next,sites,r
      real radius(mmaxtr),center(2,mmaxtr)
      real radv(mmaxtr),iord(mmaxtr)
      common /work1/ radius,center, radv,iord
      integer systeme

      common /lider/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipik,ipog
      open(1,file='rescales',status='old',form='unformatted')
      open(2,file='tempor',status='unknown',form='unformatted')
      open(7,file='triang')
      open(4,file='leader')
      open(8,file='text')
      call glider
       do 10 i1=1,maxst-10
          read (1,end=1020,err=1010) r(1,i1),r(2,i1),r(3,i1)
c         print *, r(1,i1),r(2,i1),r(3,i1)
   10     continue
      print *,'===> dimension maxst too small    <===*
      stop
 1010 print *,*   ===>  error reading        <===*
      write(8,*)'error reading*
      stop
 1020 nst=i1-1
 1030 continue
      write(8,*)'40 30*
      print *,*   ===>',nst,* points have been read      <===*
      write(8,13) nst,* points have been read*
   13 format(' *,15,a)

      call triang(r,next,sites,radius,center)
        call chunks(r,nst)

      print *,* triangulation of initial points is constructed*
      write(8,*) 'triangulation of initial points is constructed'
      idri=idra
      if(idra.le.0) then
         print *, * draw the triangulation *'
               call getans(idri)
         endif
      if(rdmax0.lt.0) then
         print *,' radius of confidence ?*
         read *,rdmax0
         endif
      rdmax=rdmax0**2
      definition of radius of confidence
         if(rdmax.le.1e-50) then
             call detrad(rdmax,radius,radv,iord)
             rdmax0=sqrt(rdmax)
             endif
         ind5 = 0
       call output(r,next,sites,radius,iord,ihardco)
       close(8)
       call plider(' triang ')
c      print *,'====>   end of job   <====*
       if (ind .eq.1) call tryl("text")
       ind5 = 1
       close(7)
        close(1)
       close(2)
       return
       end



       subroutine triang(r,next,sites,radius,center)
       common /points/ nst,ntr
       real r(3,1)
       integer next(3,1),sites(3,1)
       real radius(1),center(2,1)
       integer tgl
       logical ok
*
       nst00=nst
       call initt(r,next,sites,radius,center)


       tgl=1
       ndp=0
       do 20 np=2,nst00
           if(nst.ge.1000.and.mod(np,1000).eq.0) then
c              print *,'   ===>',np,* points have been included  <===*
               end if
*          bkl'~itx to~ku 'np' b tpiahgulqci'. dlq !togo
*          hajti tp~k 'tgl', b kotopom levit 'np'
           call search(tgl,np,ok,mk,r,next,sites)
           if(ok) then
*              cdelatx bapicehtpi~eckoe podpazdelehie
               call bar(tgl,np,mk,r,next,sites,radius,center)
*              pepectpoitx tpiahgulqci'
               call recon(tgl,r,next,sites,radius,center)
            else
               ndp=ndp+1
               end if
   20      continue
       if(ndp.ne.0) then
c         print *,'===> *,ndp,'points has not been included      <===*
c         print *,'===> check, if you execute program           <===*
c         print *,'===>   snake properly                        <===*
          write(8,*)'check,if you execute program *
          write(8,*)'snake properly*
          endif
       return
       end

        subroutine initt(r,next,sites,radius,center)
*
*      ____________
*      ihicializaciq
*      papametpy okpuvhocti, opicahhoj bokpug modeli
*      rbnd - padiuc,  nbnbp - kol~bo to~ek ha okpuvhocti
```

```
      parameter(nbndp=11,rbnd=10.)
      integer tgl
      common /points/ nst,ntr
      real r(3,1)
      integer next(3,1),sites(3,1)
      real radius(1),center(2,1)
      save

      p=atan(1.)*8
      ntr=0
      nst0=nst
      do 10 j0=1,nbndp
         ntr=ntr+1
         nst=nst+1
         r(1,nst)=rbnd*cos(p*j0/nbndp)+.5
         r(2,nst)=rbnd*sin(p*j0/nbndp)+.5
         r(3,nst)=0.0
         sites(1,j0)=nst
         sites(2,j0)=nst0+mod(j0,nbndp)+1
         sites(3,j0)=1
         next(1,j0)=mod(j0,nbndp)+1
         next(2,j0)=mod(j0+nbndp-2,nbndp)+1
         next(3,j0)=0
10    continue
      do 20 ijk=1,nbndp
         call circle(ijk,r,sites,radius,center)
20    continue
      return
      end

      subroutine bar(trl,st,mk,r,next,sites,radius,center)
      ____________________
      to-ka 'st' pome}aetcq b tp-k 'trl' ppi pomo}i
      bapicehtpi~eckogo podpazdelehiq
      common /points/ nst,ntr
      real r(3,1)
      integer next(3,1),sites(3,1)
      real radius(1),center(2,1)
      integer np,trl,tr2,tr3,tra,trb,st,stl,st2,st3
      save

      if(mk.gt.0) then
         n0=next(mk,trl)
         m0=mark(n0,trl,next)
         endif
      tr2=next(2,trl)
      tr3=next(3,trl)

      m2=mark(tr2,trl,next)
      m3=mark(tr3,trl,next)
     'tra' i 'trb' - dba novyx tpeugolxhika
      tra=ntr+1
      trb=ntr+2
      ntr=ntr+2
      if(tr2.ne.0) next(m2,tr2)=tra
      if(tr3.ne.0) next(m3,tr3)=trb
      next(2,trl)=tra
      next(3,trl)=trb

      next(1,tra)=tr2
      next(2,tra)=trb
      next(3,tra)=trl

      next(1,trb)=tr3
      next(2,trb)=trl
      next(3,trb)=tra

      st1=sites(1,trl)
      st2=sites(2,trl)
      st3=sites(3,trl)

      sites(1,trl)=st
      sites(1,tra)=st
      sites(2,tra)=st3
      sites(3,tra)=st1
      sites(1,trb)=st
      sites(2,trb)=st1
      sites(3,trb)=st2
*fllm

      if(mk.gt.0) then
         n1=next(m0,n0)
         if(next(1,n1).ne.n0) stop 33
         call flip(n1,1,next,sites,r)
         call circle(n0,r,sites,radius,center)
         endif
      return
      end

      subroutine search(tgl,np,ok,mk,r,next,sites)
*     ____________________
*     poick tpeugolxhika b kotopyj popadaet to-ka np
      parameter (e=1e-12
     .          ,el=1e-10)
      integer tgl,m,np,prev,ndp/0/
      common /points/ nst,ntr
      real r(3,1)
      integer next(3,1),sites(3,1)
      logical ok
      save
      mpr=10

      mk=0
      a3=volum(r(1,np),r(1,sites(1,tgl)),r(1,sites(2,tgl)))
      a1=volum(r(1,np),r(1,sites(2,tgl)),r(1,sites(3,tgl)))
      a2=volum(r(1,np),r(1,sites(3,tgl)),r(1,sites(1,tgl)))
      if(a1.ge.0.and.a2.ge.0.and.a3.ge.0) then
          v1=abs(a1)
          v2=abs(a2)
          v3=abs(a3)
          if(v1.le.e.or.v2.le.e.or.v3.le.e) then
              if(v1.le.e.and.v2.le.e.and.v3.le.e) then
                  ok=.false.
                  return
                  endif
              if(v1.le.e) then
                  if(v2.le.e) then
                      ok=.false.
                      ndp=ndp+1
                  else
                      if(v3.le.e) then
                          ok=.false.
                          ndp=ndp+1
                      else
                         mk=1
                         ok=.true.
                         end if
                      endif
```

```
            else
                if(v2.gt.e) then
                    mk=3
                    ok=.true.
                else
                    if(v3.le.e) then
                        ok=.false.
                        ndp=ndp+1
                    else
                       mk=2
                       ok=.true.
                       endif
                    endif
                endif
        else
            ok=.true.
            end if
        return
        end if
    if(a2.lt.a3) then
        if(a1.lt.a2) then
            m=1
            a=a1
        else
            m=2
            a=a2
            end if
    else
        if(a1.lt.a3) then
            m=1
            a=a1
        else
            m=3
            a=a3
            end if
        end if

    k0=0
10 prev=tg1
    tg1=next(m,tg1)
    m2=mark(tg1,prev,next)
    m3=mod(m2,3)+1
    m1=mod(m3,3)+1
    a3=volum(r(1,np),r(1,sites(m1,tg1)),r(1,sites(m2,tg1)))
    a1=volum(r(1,np),r(1,sites(m2,tg1)),r(1,sites(m3,tg1)))

       a0=a
    if(a1.lt.a3) then
        m=m1
        a=a1
    else
        m=m3
        a=a3
        end if
    k0=k0+1
    if(k0.gt.ntr ) then
        print *,' ===> triangle not found     <===='
        write(8,*)'fatal error'
        print *,' ===>  fatal error           <===='
        ndp=ndp+1
        ok=.false.
        return
        end if
    if(a.lt.0) goto 1010
```

```
    v1=abs(a1)
    v2=abs(a0)
    v3=abs(a3)
    if(v1.le.e.or.v2.le.e.or.v3.le.e) then
        if(v1.le.e.and.v2.le.e.and.v3.le.e) then
            ok=.false.
            ndp=ndp+1
            endif
        if(v1.le.e) then
            if(v2.le.e) then
                ok=.false.
                ndp=ndp+1
            else
                if(v3.le.e) then
                    ok=.false.
                    ndp=ndp+1
                else
                    mk=m1
                    ok=.true.
                    end if
                end if
        else
            if(v2.gt.e) then
                mk=m3
                ok=.true.
            else
                if(v3.le.e) then
                    ok=.false.
                    ndp=ndp+1
                 else
                     mk=m2
                     ok=.true.
                     endif
                endif
            endif
    else
        ok=.true.
        mk=0
        end if

        if (mk.eq.0) return
       kg1=next(mk,tg1)
       j1=mark(kg1,tg1,next)
       j2=mod(j1,3)+1
       j3=mod(j2,3)+1
    v4=abs(volum(r(1,np),r(1,sites(j1,kg1)),r(1,sites(j2,kg1))))
    v5=abs(volum(r(1,np),r(1,sites(j3,kg1)),r(1,sites(j1,kg1))))
    if(v4.lt.e.or.v5.lt.e) then
                    ok=.false.
                    ndp=ndp+1
                    endif
    return
    end

    subroutine recon(tg1,r,next,sites,radius,center)
*   ------------------------
*   pepectpojka pe{etki.
*   bocctahoblehie tpiahgulqcii delohe
*   ppi pomo}i kpugobogo kpitepiq
    common /points/ nst,ntr
    real r(3,1)
    integer next(3,1),sites(3,1)
    real radius(1),center(2,1)
```

```
      integer tgl,nxt,ngb,stp
      logical criter
      save

      nxt=tgl
      stp=0
1010  ngb=next(1,nxt)
      if(ngb.ne.0) then
         if(sites(1,nxt).ne.sites(1,tgl)) then
            print *,'    ===> "next" and "sites" are in disaccord <===='
            print *,'    ===>      fatal error                    <===='
            write(8,*)'fatal error'
            stop 41
            end if
                        call circle(ngb,r,sites,radius,center)
         if(.not.criter(ngb,sites(1,tgl),r,radius,center)) then
            call flip(nxt,1,next,sites,r)
            goto 1010
            end if
         end if
      call circle(nxt,r,sites,radius,center)
      nxt=next(2,nxt)
      stp=stp+1
      if(stp.gt.3*nst)  then
          print *,'    ===>  can"t constract triangulation   <===='
            write(8,*)'fatal error'
          return
          end if
      if(nxt.ne.tgl) goto 1010
      return
      end

      logical function criter(tg,st,r,radius,center)
```

```
ppobepka kpugobogo kpitep
popadaet li to~ka st b okpuvhoctx, opicahhu' bokpug tp-ka tg ?
ecli popadaet, to criter = .false.
real r(3,1)
real radius(1),center(2,1)
integer tg,st
save
if(tg.eq.0) stop 44
if(scal(r(1,st),center(1,tg)).lt.radius(tg)) then
    criter=.false.
else
    criter=.true.
    end if
return
end

subroutine flip(tga,ma1,next,sites,r)
```

```
common /points/ nst,ntr
integer next(3,1),sites(3,1)
integer tga,tgb,sh1
real r(3,1)
save

sh1(m)=mod(m,3)+1

ma2=sh1(ma1)
ma3=sh1(ma2)
tgb=next(ma1,tga)
if(tga.eq.0) stop 45
      mb1=mark(tgb,tga,next)
      mb2=sh1(mb1)
      mb3=sh1(mb2)
*film
c      call vector(r,sites(ma2,tga),sites(ma3,tga),5,4)
c       if(nst.lt.5)    call sleep(1)
c      call vector(r,sites(ma2,tga),sites(ma3,tga),4,13)
c      call vector(r,sites(ma1,tga),sites(mb1,tgb),5,4)
cif(nst.lt.5)   call sleep(1)
c      call vector(r,sites(ma1,tga),sites(mb1,tgb),11,7)
c       call vector(r,sites(ma1,tga),sites(ma2,tga),11,7)
c       call vector(r,sites(ma3,tga),sites(ma1,tga),11,7)
c       call vector(r,sites(mb2,tgb),sites(mb1,tgb),11,7)
c       call vector(r,sites(mb3,tgb),sites(mb1,tgb),11,7)
c       if(nst.lt.5)    call sleep(1)
*film
      if(next(ma2,tga).ne.0) then
         next(mark(next(ma2,tga),tgb,next),next(ma2,tga))=tgb
         end if
      if(next(mb2,tgb).ne.0) then
         next(mark(next(mb2,tgb),tgb,next),next(mb2,tgb))=tga
         end if
      next(ma1,tga)=next(mb2,tgb)
      next(ma1,tgb)=next(mb3,tgb)
      next(mb2,tgb)=next(ma2,tga)
      next(ma2,tga)=tgb
      next(ma3,tgb)=tga
      sites(ma3,tga)=sites(mb1,tgb)
      sites(ma3,tgb)=sites(mb2,tgb)
      sites(ma1,tgb)=sites(ma1,tga)
      sites(ma2,tgb)=sites(ma3,tga)
      return
      end


      subroutine circle(tgl,r,sites,radius,center)
*     ____________________________________
*     by~iclq'tcq koopdihaty cehtpa i padiuc okpuvhocti,
*     opicahhoj  bokpug tp-ka tgl
      real r(3,1)
      integer sites(3,1)
      real radius(1),center(2,1)
      integer tgl,st1,st2,st3
      real*8 a(2,2),b(2),det,pp
      save
      pp(x)=dprod(x,x)

      st1=sites(1,tgl)
      st2=sites(2,tgl)
      st3=sites(3,tgl)
      a(1,1)=r(1,st2)-r(1,st1)
      a(1,2)=r(2,st2)-r(2,st1)
      a(2,1)=r(1,st3)-r(1,st1)
      a(2,2)=r(2,st3)-r(2,st1)
      b(1)=(pp(r(1,st2))+pp(r(2,st2))-pp(r(1,st1))-pp(r(2,st1)))/2
      b(2)=(pp(r(1,st3))+pp(r(2,st3))-pp(r(1,st1))-pp(r(2,st1)))/2
      det=a(1,1)*a(2,2)-a(2,1)*a(1,2)
      center(1,tgl)=(a(2,2)*b(1)-a(1,2)*b(2))/det
      center(2,tgl)=(a(1,1)*b(2)-a(2,1)*b(1))/det
      radius(tgl)=pp(r(1,st1)-center(1,tgl))
     .           +pp(r(2,st1)-center(2,tgl))
      return
      end
```

triang.f

```
      real function scal(x,y)
*
*     ---------------------
      scal = (x,y)
      real x(2),y(2)
      scal=dprod(x(1)-y(1),x(1)-y(1))+dprod(x(2)-y(2),x(2)-y(2))
      return
      end

      real function volum(x,y,z)
*
*     ---------------------
      ploiadx co zhakom
*     volume = (y-x,z-x)
      real x(2),y(2),z(2)

      volum=dprod(y(1)-x(1),z(2)-x(2))-dprod(y(2)-x(2),z(1)-x(1))
      return
      end

      integer function mark(tr1,tr2,next)
      integer tr1,tr2
      integer next(3,1)

*     oppedelehie oplentacii b tp-ke tr1 po tp-ku tr2
      if(tr1.eq.0) then
         mark=0
         return
         end if
      do 10 mark=1,3
          if(next(mark,tr1).eq.tr2) return
10     continue
      print *,'     ===>  "next" is in disaccord       <===='
      print *,'     ===>       fatal error            <===='
            write(8,*)'fatal error'
      stop 51
      end

      subroutine output(r,next,sites,radius,iord,ihardco)
      bybod dlq bizualizacii i ihteppolqcii
      common /points/ nst,ntr
      real r(3,1)
      integer next(3,1),sites(3,1)
      real radius(1),center(2,1)
      logical frame
      integer iord(1)

      common /flder/ ifo,isu,nmi0,xmax,xmin,ymax,ymin,zmax,zmin,grid,
     &               madst,rdmax0,idra,iwin,idrm,msav,ilg,miner,kva
     &              ,ipix,ipog
       save
      ntr7=0
      rdmax=rdmax0**2
      do 20 nt=1,ntr
      if(radius(nt).gt.rdmax) goto 20
               if(frame(sites(1,nt),r)) goto 20
               if(frame(sites(2,nt),r)) goto 20
               if(frame(sites(3,nt),r)) goto 20

      ntr7=ntr7+1
      iord(ntr7)=nt
20    continue
      write (7,*) ntr7,nst
      do 10 j2=1,nst
          if(r(1,j2) .ge. 0.0 .and. r(1,j2).le. 1.0) then
          if(r(2,j2) .ge. 0.0 .and. r(2,j2).le. 1.0) then
             write (7,*) r(1,j2),r(2,j2),r(3,j2)
             else
             write(7,*)r1,r1,r(3,j2)
             end if
             else
             write(7,*)r1,r1,r(3,j2)
             end if
10     continue
       do 710 i3=1,ntr7
          j2=iord(i3)
          write (7,*) sites(1,j2),sites(2,j2),sites(3,j2)
710    continue
       if(madst.eq.0.and.isu.eq.1) return
       write (2) ntr,nst
       write (2) ((sites(i0,k0),i0=1,3),k0=1,ntr)
       write (2) ((next(i0,k0),i0=1,3),k0=1,ntr)
       if(ihardco .eq.1) then
        call hardcopy(r,sites,filename)
       end if
       return
       end

       subroutine detrad(rdmax,radius,radv,iord)
       common /points/ nst,ntr
       real radius(1)
       real radv(1),iord(1)
          nnt=0
           do 30 i2=1,ntr
               if(radius(i2).gt.5.) goto 30
               radv(i2)=radius(i2)
               nnt=nnt+1
30             continue
          call kb07a(radv,nnt,iord)
*      definition of radius of confidence
       rr=0
       nt=nnt*.9
       do 10 ktr=1,nt
       rr=rr+radv(ktr)
10     continue
       rdmax=rr/nt
*      rdmax=rr/nt*10
c      print *,'radius of confidence =',sqrt(rdmax)
       return
       end

       subroutine hardcopy(r,sites)
       common /points/ nst,ntr
       real r(3,1)
       integer sites(3,1)
       real x(3),y(3)
       mm(i)=mod(i,3)+1
       open(10,file='hardtmp')
       write(10,*) 'limits 0 1 0 1'
       write(10,*) 'lweight 3'
       write(10,*) 'ptype 10  3'
       write(10,*) 'erase'
       do 10 nt=1,ntr
          do 30 i=1,3
         m1=sites(       mm(i),nt)
         m2=sites(       mm(i+1),nt)
          if(m1.lt.m2) goto 30
          if(r(1,m1).lt.0.or.r(1,m1).gt.1) goto 30
          if(r(2,m1).lt.0.or.r(2,m1).gt.1) goto 30
          if(r(1,m2).lt.0.or.r(1,m2).gt.1) goto 30
```

```
        if(r(2,m2).lt.0.or.r(2,m2).gt.1) goto 30
      write(10,*)'relocate' ,r(2,m1),r(1,m1)
      write(10,*)'draw' ,r(2,m2),r(1,m2)
 30   continue
 10   continue
      close(10)
       return
      end



c ------------------------------------------------------------s
      logical function bad(n,rmin,radius,center)
      parameter(rvmin=1e-8,rvmax=0.7)
      logical cirin
      real radius(1),center(2,1)
c***  begin
      bad=radius(n).lt.rmin
      if(bad) return
      bad=radius(n).ge.rvmax
      if(bad) return
c **  circle inside?
      bad=.not.cirin(n,circle)
      return
c **  end function
      end
c ------------------------------------------------------------s
      logical function cirin( nt,circle )
      real circle(2,1)
c **  begin cirin

      cirin=abs(circle(1,nt)-.5).lt..6
      cirin=cirin.and.(abs(circle(2,nt)-.5).lt..6)
      return
c **  end function cirin
      end
c ------------------------------------------------------------s
      logical function border(n,r,sites,radius)
      parameter(rvmin=1e-8,rvmax=0.7)
      logical frame
      integer sites(3,1)
      real r(3,1),radius(1)
c **  begin
      border=radius(n).ge.rvmax
      if(border) return
      ns1=sites(1,n)
      border=frame(ns1,r)
      if(border) return
      ns2=sites(2,n)
      border=frame(ns2,r)
      if(border) return
      ns3=sites(3,n)
      border=frame(ns3,r)
      return
c **  end function
      end
c ------------------------------------------------------------s
      subroutine distance(r,sites,np,tgl,rad,kmin)
      real r(3,1),min
      integer sites(3,1)
      integer tgl

      do 40 k = 1,3
      dist =        (r(1,np) - r(1,sites(k,tgl))) ** 2
      dist = dist + (r(2,np) - r(2,sites(k,tgl))) ** 2
        if(k.eq.1) then
                dmin=dist
                kmin=1
                else if(dist.lt.dmin) then
                        kmin=k
                        dmin=dist
                        endif
 40   continue
        if(dmin.gt.rad) kmin=-1

      return
      end
```

```
define PATH "/usr/people/lila/FILM/"
define PLACE "< triang &"
```

```
/*
 *      event.h
 * External interface and defines to input-queue event handling
 * routines.
 * Written by Wade Olsen for Silicon Graphics, Inc.
 */

/*
 *      The event handler understands two kinds of things; events and
 * updates.  Events are reactions to things occuring in the input
 * queue.  Updates are functions that should be called whenever there
 * is nothing waiting in the input queue, and may be active or
 * inactive.  If there are no active updates and nothing in the input
 * queue, then event() will block, using up no CPU time.
 *
 * add_event is used to look for events.  The first three arguments
 * are used to identify which event to look for.  The first argument
 * is the window (gid) the event must happen in; if this value is ANY
 * then any window will do.  The second argument is the device to look
 * for (e.g.  RIGHTMOUSE or REDRAW or KEYBD, etc).  Again, if it is
 * ANY then any device will match.  The third argument is the value
 * the device must generate (e.g.  DOWN or UP); ANY means all values
 * match.  The last two arguments are what should be done when an
 * event is generated.  The fourth argument is a function to be
 * called, and the fifth is an argument that should be supplied to the
 * function.  In addition, the value generated by the device will also
 * be passed to the function when it is called.
 *
 * For example,
 *      add_event(winget(), RIGHTMOUSE, DOWN, dopup, my_menus);
 *      qdevice(RIGHTMOUSE);
 * will make a pop-up menu appear when the right mousebutton goes
 * down.  Note that you must do the qdevice() call yourself.
 */
void add_event(int, int, int, void (*fn)(void *, int), char *) ;

/*
 * An update is like an event, only simpler.  The first argument is a
 * pointer to an integer flag specifying whether or not this update
 * function is active.  The second is a function to be called when it
 * is active, and the last is an argument to be supplied to the
 * function.
 */
void add_update(int *, void (*fn)(void *), char *) ;

/*
 * Finally, when all updates and events have been added, repeatedly
 * call event() to handle them -- something like
 *
 *      while (quitflag == FALSE) event();
 *
 * You should have previously added an event that sets quitflag to
 * TRUE, of course.
 */
void event(void) ;

/*
 *      These are some useful defines for the possible values buttons
 * can generate.
 */
#define ANY     -1
#define UP      0
#define DOWN    1

/*
 *      And a few external variables you might find useful
 */
extern int context, state, device;
```

```
/*
 *      This software is in the public domain, it may not be resold
 *      or relicensed.  Modified and enhanced versions of this software
 *      are likewise to be made freely available.  Sites using this
 *      software are requested to register with NASA at the address below.
 *      Send modifications and requests for most recent version to:
 *
 *      Author:  David A. Tristram
 *               ATTN: Panel Library
 *               MS 258-5
 *               NASA  Ames Research Center
 *               Moffett Field, CA  94035
 *
 *               415-694-4404
 *               dat@orville.nas.nasa.gov
 */

* external functions */

:ifdef lint
:ifdef IRIS_4D
ong lseek(int, long, int);
ar *calloc(unsigned, unsigned);
oid exit(int);
oid perror(char *);
t write(int, char *, unsigned);
t read(int, char *, unsigned);
else  IRIS_4D
ong lseek();
ar *calloc();
oid exit();
oid perror();
t write();
t read();
endif IRIS_4D
endif lint

* constants */

.fndef PNL_EDITOR_PARSING
:ifndef PNL_EXTERN
efine PNL_EXTERN  extern
ndif
.fndef PNL_INIT
efine PNL_INIT(x) /* nothing */
ndif

.f 0
.fndef FILE
.nclude <stdio.h>
ndif
.nclude <string.h>
.nclude <sys/types.h>
ndif

.fdef TRUE
.ndef TRUE
ndif TRUE
efine TRUE    1
 fndef NULL
efine NULL    0
ndif
 fndef forever
efine forever for (;;)
#endif
#ifndef PI
#define PI      3.14159265358979323846      /* from math.h */
#endif
#define PNL_RAD_TO_DEG  (0.31830988618379067154*180.0)
#define PNL_TABLE_SIZE  4096            /* max number of panels + actuators */

#ifndef MAXSGIDEVICE
#define MAXSGIDEVICE    20000
#endif  MAXSGIDEVICE
#define PNL_DEVICE_OFFSET   (MAXSGIDEVICE+1)
#define PNL_TOKEN           (PNL_DEVICE_OFFSET+0)

#define PNL_PIXELS_PER_UNIT     50.0 /* used to size the panel */
#define PNL_CHAR_THRESHOLD      35.0 /* ppu below this, draw chars as rects */
#define PNL_PPU_MIN_DELTA       0.5  /* smallest change to cause recalc */

#define PNL_CHAR_PIXEL_HEIGHT   10
#define PNL_CHAR_PIXEL_WIDTH    10

#ifdef IRIS_NEWS
#  define PNL_SCREEN_BORDER     12
#  define PNL_TITLE_BAR_HEIGHT  19
#else
#  define PNL_SCREEN_BORDER     6
#  define PNL_TITLE_BAR_HEIGHT  18
#endif

/* panel dimensions */

#define PNL_MARGIN              0.2     /* absolute edge of panel left blank */
#define PNL_DIM_1               0.2     /* absolute dist tween actuators */
#define PNL_DIM_2               0.1     /* absolute dist to a label */
#define PNL_DIM_3               0.05    /* a tiny little distance */
#define PNL_DIM_4               0.025   /* half of THAT! */
#define PNL_BEVEL_WIDTH         PNL_DIM_3

/* actuator dimensions */

#define PNL_SLIDER_HEIGHT       (6.0-PNL_DIM_1) /* w/o label */
#define PNL_SLIDER_WIDTH        (1.0-PNL_DIM_1)
#define PNL_SLIDER_BAR_HEIGHT   (0.05*PNL_SLIDER_HEIGHT)
#define PNL_DIFFERENTIAL_FACTOR 0.1
#define PNL_FINE_CONTROL_FACTOR 0.05
#define PNL_BUTTON_EDGE         (0.5-0.5*PNL_DIM_1)
#define PNL_WIDE_BUTTON_WIDTH   (2.0-PNL_DIM_1)
#define PNL_WIDE_BUTTON_HEIGHT  PNL_BUTTON_EDGE
#define PNL_TYPEIN_LENGTH       40  /* default length in characters */
#define PNL_TYPEIN_MAX_STRING   256
#define PNL_TYPEOUT_BUFSIZ      4096
#define PNL_TYPEOUT_COLUMNS     PNL_TYPEIN_LENGTH
#define PNL_TYPEOUT_LINES       4
#define PNL_TYPEOUT_MAX_COLUMNS 256
#define PNL_SCROLLBAR_WIDTH     (2.0*PNL_DIM_1)
#define PNL_TYPEOUT_ARROW_HEIGHT    (PNL_SCROLLBAR_WIDTH/2.0)
#define PNL_PUCK_EDGE           (4.0-PNL_DIM_1)
#define PNL_PUCK_SIZE           (1.0-PNL_DIM_1)
#define PNL_FLOATING_PUCK_EDGE  (1.0-PNL_DIM_1)
#define PNL_FLOATING_PUCK_SENS  10.0    /* units per full range output */
#define PNL_METER_HEIGHT        (2.0-PNL_DIM_1)
#define PNL_METER_WIDTH         (3.0-PNL_DIM_1)
#define PNL_METER_LINEWIDTH     4 /* Pixels for meter pointer and bar line */
#define PNL_STRIP_HEIGHT        (2.0-PNL_DIM_1)
#define PNL_STRIP_WIDTH         (4.0-PNL_DIM_1)
```

```
#define PNL_STRIP_LINEWIDTH     2
#define PNL_STRIP_CHART_NPTS    200
#define PNL_SLIDEROID_HEIGHT    (1.5-PNL_DIM_1)
#define PNL_SLIDEROID_WIDTH     (2.0-PNL_DIM_1)
#define PNL_DIAL_EDGE           (1.0-PNL_DIM_1)
#define PNL_DIAL_WINDS          0.88    /* revs per full range output */
#define PNL_MULTISLIDER_DIVISIONS 5
#define PNL_MENU_WIDTH          PNL_WIDE_BUTTON_WIDTH
#define PNL_MENU_TITLE_HEIGHT   PNL_WIDE_BUTTON_HEIGHT
 define PNL_ICON_WIDTH          (2.0-PNL_DIM_1)
 define PNL_ICON_HEIGHT         (0.5-PNL_DIM_2)
 define PNL_SCROLL_WIDTH        (4.0-PNL_DIM_1)
 define PNL_SCROLL_HEIGHT       (6.0-PNL_DIM_1)

 * Actuator types */

 define PNL_MAXACT              0x7ffff     /* user types start here+1 */
 define PNL_USER_OFFSET         0x80000     /* first user actuator */

 define PNL_SLIDER              0
 define PNL_VSLIDER             PNL_SLIDER
 define PNL_HSLIDER             1
 define PNL_FILLED_SLIDER       2
 define PNL_FILLED_VSLIDER      PNL_FILLED_SLIDER
 define PNL_FILLED_HSLIDER      3
 define PNL_DVSLIDER            4
 define PNL_DHSLIDER            5
 define PNL_BUTTON                      310
 define PNL_TOGGLE_BUTTON               311
 define PNL_RADIO_BUTTON                312
 define PNL_WIDE_BUTTON                 313
 define PNL_SHADOW_BUTTON               314
 define PNL_LEFT_ARROW_BUTTON           315
 define PNL_RIGHT_ARROW_BUTTON          316
 define PNL_UP_ARROW_BUTTON             317
 define PNL_DOWN_ARROW_BUTTON           318
 define PNL_LEFT_DOUBLE_ARROW_BUTTON    319
 define PNL_RIGHT_DOUBLE_ARROW_BUTTON   320
 define PNL_UP_DOUBLE_ARROW_BUTTON      321
 define PNL_DOWN_DOUBLE_ARROW_BUTTON    322
 define PNL_TYPEIN              20
 define PNL_LABEL               21
 define PNL_TYPEOUT             30
 define PNL_METER               40
 define PNL_ANALOG_METER        PNL_METER
 define PNL_ANALOG_BAR          41
 define PNL_STRIP_CHART         42
 define PNL_SCALE_CHART         43
 define PNL_PUCK                50
 define PNL_FLOATING_PUCK       51
 define PNL_RUBBER_PUCK         52
 define PNL_SLIDEROID           60
 define PNL_PALETTE             70
 define PNL_DIAL                80
 define PNL_MULTISLIDER         90
 define PNL_VMULTISLIDER        PNL_MULTISLIDER
 define PNL_HMULTISLIDER        91
 define PNL_MULTISLIDER_BAR             96
 define PNL_MULTISLIDER_OPEN_BAR        97
 define PNL_VMULTISLIDER_BAR            PNL_MULTISLIDER_BAR
 define PNL_VMULTISLIDER_OPEN_BAR       PNL_MULTISLIDER_OPEN_BAR
 define PNL_HMULTISLIDER_BAR            98
 define PNL_HMULTISLIDER_OPEN_BAR       99
 define PNL_MENU                100

#define PNL_MENU_ITEM           110
#define PNL_ICON                120
#define PNL_ICON_MENU           130
#define PNL_SUB_MENU            132
#define PNL_FRAME               140
#define PNL_CYCLE               150
#define PNL_SCROLL              160
#define PNL_MOUSE               1000

/* slider modes */

#define PNL_SM_NORMAL           0x0
#define PNL_SM_DIFFERENTIAL     0x1
#define PNL_SM_FINE_CONTROL     0x2
#define PNL_SM_NOSNAP           0x4

/* multislider modes */

#define PNL_MSM_FREE            0x00    /* these three mutually exclusive */
#define PNL_MSM_ORDERED         0x01
#define PNL_MSM_CONSTRAINED     0x02
#define PNL_MSM_ADD             0x04    /* these two mutually exclusive */
#define PNL_MSM_DELETE          0x08

/* icon modes */

#define PNL_IM_STOWED           0x01
#define PNL_IM_OPEN             0x02

/* typeout modes */

#define PNL_TOM_NORMAL          0x00
#define PNL_TOM_NOCURSOR        0x01
#define PNL_TOM_NOREGION        0x02

/* frame modes */

#define PNL_FM_FREE             0x00    /* these three mutually exclusive */
#define PNL_FM_FIXED            0x01
#define PNL_FM_FIXED_SIZE       0x02

/* cycle modes */

#define PNL_CM_NORMAL           0x00
#define PNL_CM_UNDER_CONSTRUCTION       0x01

/* drawing styles */

#define PNL_OPEN                0
#define PNL_FILLED              1

/* label placement (anti-clockwise from right) */

#define PNL_LABEL_RIGHT         0
#define PNL_LABEL_RIGHT_TOP     1
#define PNL_LABEL_UPPER_RIGHT   2
#define PNL_LABEL_TOP_RIGHT     3
#define PNL_LABEL_TOP           4
#define PNL_LABEL_TOP_LEFT      5
#define PNL_LABEL_UPPER_LEFT    6
#define PNL_LABEL_LEFT_TOP      7
#define PNL_LABEL_LEFT          8
#define PNL_LABEL_LEFT_BOTTOM   9
#define PNL_LABEL_LOWER_LEFT    10
```

```
#define PNL_LABEL_BOTTOM_LEFT   11
#define PNL_LABEL_BOTTOM        12
#define PNL_LABEL_BOTTOM_RIGHT  13
#define PNL_LABEL_LOWER_RIGHT   14
#define PNL_LABEL_RIGHT_BOTTOM  15
#define PNL_LABEL_CENTER        16
#define PNL_LABEL_NORMAL        17      /* no offset, actually quite abnormal */

#define PNL_MAX_LABEL_TYPE      17

/* colors */

#define PNL_RGB_WHITE_COLOR             255, 255, 255   /* RGB triples */
#define PNL_RGB_BEVEL_LIGHT_COLOR       209, 236, 255
#define PNL_RGB_NORMAL_COLOR            125, 165, 223
#define PNL_RGB_OTHER_COLOR              62, 116, 207
#define PNL_RGB_BACKGROUND_COLOR         88, 134, 213
#define PNL_RGB_HIGHLIGHT_COLOR           0,  65, 191
#define PNL_RGB_BEVEL_DARK_COLOR         19,  42,  80
#define PNL_RGB_LABEL_COLOR               0,   0,  60
#define PNL_RGB_BLACK_COLOR               0,   0,   0

#define PNL_12BIT_WHITE_COLOR           30              /* map indices */
#define PNL_12BIT_BEVEL_LIGHT_COLOR     229
#define PNL_12BIT_NORMAL_COLOR          178
#define PNL_12BIT_BACKGROUND_COLOR      173
#define PNL_12BIT_OTHER_COLOR           147
#define PNL_12BIT_HIGHLIGHT_COLOR       96
#define PNL_12BIT_BEVEL_DARK_COLOR      66
#define PNL_12BIT_LABEL_COLOR           36
#define PNL_12BIT_BLACK_COLOR           31

#define PNL_8BIT_WHITE_COLOR            30
#define PNL_8BIT_BEVEL_LIGHT_COLOR      29
#define PNL_8BIT_NORMAL_COLOR           23
#define PNL_8BIT_BACKGROUND_COLOR       20
#define PNL_8BIT_OTHER_COLOR            18
#define PNL_8BIT_HIGHLIGHT_COLOR        13
#define PNL_8BIT_BEVEL_DARK_COLOR       11
#define PNL_8BIT_LABEL_COLOR            9
#define PNL_8BIT_BLACK_COLOR            8

#define PNL_4BIT_WHITE_COLOR            WHITE
#define PNL_4BIT_BEVEL_LIGHT_COLOR      15
#define PNL_4BIT_NORMAL_COLOR           14
#define PNL_4BIT_BACKGROUND_COLOR       13
#define PNL_4BIT_OTHER_COLOR            12
#define PNL_4BIT_HIGHLIGHT_COLOR        11
#define PNL_4BIT_BEVEL_DARK_COLOR       10
#define PNL_4BIT_LABEL_COLOR            9
#define PNL_4BIT_BLACK_COLOR            BLACK

PNL_EXTERN Colorindex pnl_white_color       PNL_INIT(0);
PNL_EXTERN Colorindex pnl_bevel_light_color PNL_INIT(0);
PNL_EXTERN Colorindex pnl_normal_color      PNL_INIT(0);
PNL_EXTERN Colorindex pnl_background_color  PNL_INIT(0);
PNL_EXTERN Colorindex pnl_other_color       PNL_INIT(0);
PNL_EXTERN Colorindex pnl_highlight_color   PNL_INIT(0);
PNL_EXTERN Colorindex pnl_bevel_dark_color  PNL_INIT(0);
PNL_EXTERN Colorindex pnl_label_color       PNL_INIT(0);
PNL_EXTERN Colorindex pnl_black_color       PNL_INIT(0);

* mouse action sources */

#define PNL_SRC_QUEUE   0
#define PNL_SRC_SCRIPT  1

/* script message types */

#define PNL_MT_STATE    0xabc0
#define PNL_MT_MOUSE    0xabc1
#define PNL_MT_DELAY    0xabc2
#define PNL_MT_ENDBLOCK 0xabc3

/* function modes, ie, the context an actuator's function is called in */

#define PNL_FCNM_NONE   0
#define PNL_FCNM_DOWN   1
#define PNL_FCNM_ACTIVE 2
#define PNL_FCNM_UP     3

/* dopanel return modes, global control over dopanel return values */

#define PNL_DRM_RETURN_NULL     0
#define PNL_DRM_RETURN_PNL_CA   1

/* math relations */

#define PNL_HITRECT(x,y,x1,y1,x2,y2)    \
    ((x)>(x1) && (x)<(x2) && (y)>(y1) && (y)<(y2))
#define PNL_HITACT(a,wx,wy)                                             \
  (a->beveled?                                                          \
    PNL_HITRECT((wx),(wy),                                              \
                (a)->x-PNL_BEVEL_WIDTH,                                 \
                (a)->y-PNL_BEVEL_WIDTH,                                 \
                (a)->x+(a)->w+PNL_BEVEL_WIDTH,                          \
                (a)->y+(a)->h+PNL_BEVEL_WIDTH):                         \
    PNL_HITRECT((wx),(wy),(a)->x,(a)->y,(a)->x+(a)->w,(a)->y+(a)->h))

#define ABS(a)          ((a)>0.0?(a):-(a))
#define MIN(a, b)       ((a)<(b)?(a):(b))
#define MAX(a, b)       ((a)>(b)?(a):(b))
#define RANGE(a, b1, b2)                        \
                                                \
        ((b1)<(b2)?                             \
             ((a)<(b1)?                         \
                  (b1):                         \
                  ((a)>(b2)?                    \
                       (b2):(a))):              \
             ((a)<(b2)?                         \
                  (b2):                         \
                  ((a)>(b1)?                    \
                       (b1):(a))))

#define PNL_VAL_TO_WORLD(val, min, max, dist)   \
  (((val)-(min))/((max)-(min))*(dist))

#define PNL_WORLD_TO_VAL(x, min, max, dist)     \
  ((x)/(dist)*((max)-(min))+(min))

/* structure references */

#define PNL_ACCESS(type, ptr, member)   (((type *)(ptr->data))->member)
#define PNL_MKUSERACT(struct_type, type)                        \
    ((struct struct_type *)                                     \
    _mkuseract (sizeof (struct struct_type), type))

#endif PNL_EDITOR_PARSING
```

```
#define PNL_LABEL_BOTTOM_LEFT   11
#define PNL_LABEL_BOTTOM        12
#define PNL_LABEL_BOTTOM_RIGHT  13
#define PNL_LABEL_LOWER_RIGHT   14
#define PNL_LABEL_RIGHT_BOTTOM  15
#define PNL_LABEL_CENTER        16
#define PNL_LABEL_NORMAL        17      /* no offset, actually quite abnormal */

#define PNL_MAX_LABEL_TYPE      17

/* colors */

#define PNL_RGB_WHITE_COLOR             255, 255, 255   /* RGB triples */
#define PNL_RGB_BEVEL_LIGHT_COLOR       209, 236, 255
#define PNL_RGB_NORMAL_COLOR            125, 165, 223
#define PNL_RGB_OTHER_COLOR              62, 116, 207
#define PNL_RGB_BACKGROUND_COLOR         88, 134, 213
#define PNL_RGB_HIGHLIGHT_COLOR           0,  65, 191
#define PNL_RGB_BEVEL_DARK_COLOR         19,  42,  80
#define PNL_RGB_LABEL_COLOR               0,   0,  60
#define PNL_RGB_BLACK_COLOR               0,   0,   0

#define PNL_12BIT_WHITE_COLOR           30              /* map indices */
#define PNL_12BIT_BEVEL_LIGHT_COLOR     229
#define PNL_12BIT_NORMAL_COLOR          178
#define PNL_12BIT_BACKGROUND_COLOR      173
#define PNL_12BIT_OTHER_COLOR           147
#define PNL_12BIT_HIGHLIGHT_COLOR       96
#define PNL_12BIT_BEVEL_DARK_COLOR      66
#define PNL_12BIT_LABEL_COLOR           36
#define PNL_123IT_BLACK_COLOR           31

#define PNL_8BIT_WHITE_COLOR            30
#define PNL_8BIT_BEVEL_LIGHT_COLOR      29
#define PNL_8BIT_NORMAL_COLOR           23
#define PNL_8BIT_BACKGROUND_COLOR       20
#define PNL_8BIT_OTHER_COLOR            18
#define PNL_8BIT_HIGHLIGHT_COLOR        13
#define PNL_8BIT_BEVEL_DARK_COLOR       11
#define PNL_8BIT_LABEL_COLOR            9
#define PNL_8BIT_BLACK_COLOR            8

#define PNL_4BIT_WHITE_COLOR            WHITE
#define PNL_4BIT_BEVEL_LIGHT_COLOR      15
#define PNL_4BIT_NORMAL_COLOR           14
#define PNL_4BIT_BACKGROUND_COLOR       13
#define PNL_4BIT_OTHER_COLOR            12
#define PNL_4BIT_HIGHLIGHT_COLOR        11
#define PNL_4BIT_BEVEL_DARK_COLOR       10
#define PNL_4BIT_LABEL_COLOR            9
#define PNL_4BIT_BLACK_COLOR            BLACK

PNL_EXTERN Colorindex pnl_white_color       PNL_INIT(0);
PNL_EXTERN Colorindex pnl_bevel_light_color PNL_INIT(0);
PNL_EXTERN Colorindex pnl_normal_color      PNL_INIT(0);
PNL_EXTERN Colorindex pnl_background_color  PNL_INIT(0);
PNL_EXTERN Colorindex pnl_other_color       PNL_INIT(0);
PNL_EXTERN Colorindex pnl_highlight_color   PNL_INIT(0);
PNL_EXTERN Colorindex pnl_bevel_dark_color  PNL_INIT(0);
PNL_EXTERN Colorindex pnl_label_color       PNL_INIT(0);
PNL_EXTERN Colorindex pnl_black_color       PNL_INIT(0);

/* mouse action sources */

#define PNL_SRC_QUEUE   0
#define PNL_SRC_SCRIPT  1

/* script message types */

#define PNL_MT_STATE    0xabc0
#define PNL_MT_MOUSE    0xabc1
#define PNL_MT_DELAY    0xabc2
#define PNL_MT_ENDBLOCK 0xabc3

/* function modes, ie, the context an actuator's function is called in */

#define PNL_FCNM_NONE   0
#define PNL_FCNM_DOWN   1
#define PNL_FCNM_ACTIVE 2
#define PNL_FCNM_UP     3

/* dopanel return modes, global control over dopanel return values */

#define PNL_DRM_RETURN_NULL     0
#define PNL_DRM_RETURN_PNL_CA   1

/* math relations */

#define PNL_HITRECT(x,y,x1,y1,x2,y2)    \
    ((x)>(x1) && (x)<(x2) && (y)>(y1) && (y)<(y2))
#define PNL_HITACT(a,wx,wy)                                                     \
  (a->beveled?                                                                  \
    PNL_HITRECT((wx),(wy),                                                      \
                (a)->x-PNL_BEVEL_WIDTH,                                         \
                (a)->y-PNL_BEVEL_WIDTH,                                         \
                (a)->x+(a)->w+PNL_BEVEL_WIDTH,                                  \
                (a)->y+(a)->h+PNL_BEVEL_WIDTH):                                 \
    PNL_HITRECT((wx),(wy),(a)->x,(a)->y,(a)->x+(a)->w,(a)->y+(a)->h))

#define ABS(a)          ((a)>0.0?(a):-(a))
#define MIN(a, b)       ((a)<(b)?(a):(b))
#define MAX(a, b)       ((a)>(b)?(a):(b))
#define RANGE(a, b1, b2)                        \
                                                \
        ((b1)<(b2)?                             \
             ((a)<(b1)?                         \
                  (b1):                         \
                  ((a)>(b2)?                    \
                       (b2):(a))):              \
             ((a)<(b2)?                         \
                  (b2):                         \
                  ((a)>(b1)?                    \
                       (b1):(a))))

#define PNL_VAL_TO_WORLD(val, min, max, dist)       \
  (((val)-(min))/((max)-(min))*(dist))

#define PNL_WORLD_TO_VAL(x, min, max, dist)         \
  ((x)/(dist)*((max)-(min))+(min))

/* structure references */

#define PNL_ACCESS(type, ptr, member)     (((type *)(ptr->data))->member)
#define PNL_MKUSERACT(struct_type, type)                        \
    ((struct struct_type *)                                     \
    _mkuseract (sizeof (struct struct_type), type))

#endif PNL_EDITOR_PARSING
```

```
/* structure declarations */

typedef struct panel {
    short id;                   /* unique id  */
    struct actuator *a;         /* current actuator */
    struct actuator *al;                /* actuator list */
    struct alist *autolist;             /* list of auto actuators */
    struct actuator *lastgroup;         /* last actuator added to a group */
    Boolean active, selectable;
    long x, y, w, h;    /* screen location of the window and its size */
    Coord mx,my;    /* bounding box enclosing all actuators and labels */
    Coord maxx;
    Coord miny;
    Coord maxy;
    Coord cw,ch;    /* char width and height */
    short gid;      /* mex window number of this panels window */
    short usergid; /* mex window number of one of the user's windows */
    Object vobj;    /* viewing transformations */
    float ppu;      /* pixels per unit */
    char *label;
#ifdef IRIS_4D
    void (*delfunc)(struct panel *);
    void (*fixfunc)(struct panel *);
#else  IRIS_4D
    void (*delfunc)();
    void (*fixfunc)();
#endif IRIS_4D
    void (*drawfunc)();
    void (*downfunc)();
    void (*activefunc)();
    void (*upfunc)();
    Boolean visible;            /* crank up a window for this guy? */
    int somedirty;              /* is there a dirty act on this panel */
    int dirtycnt;
    struct panel *next;
} Panel;

typedef struct actuator {
    short id;                   /* unique id */
    struct panel *p;            /* pointer to enclosing panel */
    struct actuator *pa;        /* parent actuator */
    struct actuator *ca;        /* current subactuator (if any) */
    struct actuator *al;        /* list of sub-actuators */
    int na;                     /* number of sub-actuators */
    int type;                   /* type id, init func sets this */
    Boolean active;
    Coord x, y, w, h;
    Coord lx, ly, lw, lh;       /* offset and size of label */
    Coord ld;                   /* descender size */
    float val, extval, initval, maxval, minval;
    float scalefactor;
    char *label;
    Device key;                 /* keyboard equivalent */
    int labeltype;  /* location of the label relative to the actuator */
    void (*initfunc)();         /* the function that created me */
    void (*addfunc)();
#ifdef IRIS_4D
    void (*addsubfunc)(struct actuator *, struct actuator *);   /* to add a sub act to t
his act */
    void (*fixfunc)(struct actuator *);
    Boolean (*pickfunc)(/*struct actuator *,struct panel *,float,float*/);
    void (*delfunc)(struct actuator *);
    void (*newvalfunc)(/*struct actuator *,struct panel *,float,float*/);
    void (*dumpfunc)(struct actuator *, int);
    void (*loadfunc)(struct actuator *, int);
#else  IRIS_4D
    void (*addsubfunc)();       /* to add a sub act to this act */
    void (*fixfunc)();
    Boolean (*pickfunc)();
    void (*delfunc)();
    void (*newvalfunc)();
    void (*dumpfunc)();
    void (*loadfunc)();
#endif IRIS_4D
    void (*drawfunc)();
    void (*downfunc)();
    void (*activefunc)();
    void (*upfunc)();
    int dirtycnt;
    char *u;        /* pointer to arbitrary user data */
    char *data;     /* pointer to data peculiar to a particular actuator */
    int datasize;   /* size of data struct plus everyting it points to */
    Boolean automatic;   /* true ==> newvalfunc called every dopanel */
    Boolean selectable; /* false ==> unpickable, newvalfunc never called */
    Boolean visible; /* does this actuator have a visible manifestation? */
    Boolean beveled; /* is this actuator got a beveled edge? */
    struct actuator *group;
    struct actuator *next;
} Actuator;

#ifndef PNL_EDITOR_PARSING

typedef struct alist {          /* generic list cons cell */
    Actuator *a;
    struct alist *next;
} Alist;

typedef struct {
  float x, y;
} Point;

/* actuator specific data */

typedef struct {
  int firstpt, lastpt;
  Boolean Bind_Low, Bind_High;
  float *y;
  Actuator *lowlabel, *highlabel;
} Stripchart;

typedef struct {
  int mode;
  float finefactor;
  float differentialfactor;
  float valsave;
  Coord wsave;
  Coord bh;             /* slider bar height */
} Slider;

typedef struct {
  int mode;
  int n;                /* number of sliderbars */
  float finefactor;
  Coord wsave;
  Actuator *sa;         /* last selected sub actuator */
  Coord bh;             /* slider bar height */
  Coord clrx, clry, clrw, clrh; /* expanded label clearing area */
```

```
  void (*acttype)();    /* the init function for the slider bar act */
  Multislider;

ypedef struct {
  char *str;
  int len;
  Typein;

ypedef struct {
  int mode;
  char *buf;            /* text to be displayed */
  char *delimstr;       /* auto 'word' selction delimiters */
  int start;            /* first char to display (appears in upper-left) */
  int dot;              /* insertion point */
  int mark;             /* other end of selection region */
  int col,lin;          /* width and height in character positions */
  int len;              /* number of chars in buffer */
  int size;             /* buffer size */
  Coord ch, cw, cd;     /* character dimensions */
  Typeout;

ypedef Point Puck;

pedef struct {
  Coord x;
  Coord y;
  Mouse;

pedef struct {
  int mode;
  Boolean finemode, resetmode;
  float *resettarget, resetval;
  float valsave;
  Coord wsave;
  Slideroid;

pedef Slider Palette;

pedef struct {
  int mode;
  float finefactor;
  float valsave;
  Coord wsave;
  float winds;
  Dial;

  pedef struct {
  Coord th;     /* title height */
  Menu;

  pedef struct {
  int mode;
  Coord xstowed, ystowed, wstowed, hstowed;
  Coord xopen, yopen, wopen, hopen;
  char *labelsave;
  Icon;

  pedef struct {
  int mode;
  Coord offx, offy;     /* offset for drawing, allows scrolling */
  Coord minx;       /* bounding box enclosing all actuators and labels */
  Coord maxx;
  Coord miny;
  Coord maxy;
} Frame;        /* eventually change name to Panel? */

typedef struct {
  int mode;
  Actuator *shiftleftbutton;
  Actuator *shiftrightbutton;
  Actuator *frame;
  Alist *memberlist, *currentmember;
} Cycle;

typedef struct {
  Actuator *vslider, *hslider;
  Actuator *frame, *subframe;
} Scroll;

/* globals */

PNL_EXTERN int pnl_id           PNL_INIT(0);
                        /* an incrementing index into the table */

PNL_EXTERN char *pnl_table[PNL_TABLE_SIZE];
                        /* table of pointers to all panels and actuators */

PNL_EXTERN Panel *pnl_pl        PNL_INIT(NULL);
                        /* list of all control panels */

PNL_EXTERN Alist *pnl_kl        PNL_INIT(NULL);
                        /* a list of actuators with key equivalents */

PNL_EXTERN Panel *pnl_cp        PNL_INIT(NULL);
                        /* panel being moused (current panel) */

PNL_EXTERN Actuator *pnl_ca     PNL_INIT(NULL);
                        /* actuator being moused (current actuator) */

PNL_EXTERN Panel *pnl_cp_save   PNL_INIT(NULL);
                        /* last cp from queue */

PNL_EXTERN Actuator *pnl_ca_save        PNL_INIT(NULL);
                        /* last ca from queue */

PNL_EXTERN Boolean pnl_ca_active_save   PNL_INIT(FALSE);
                        /* state of active field in pnl_ca_save */

PNL_EXTERN Boolean pnl_cp_active_save   PNL_INIT(FALSE);
                        /* state of active field in pnl_cp_save */

PNL_EXTERN Actuator *pnl_mouse_act      PNL_INIT(NULL);
                        /* the mouse actuator, if any */

PNL_EXTERN Screencoord pnl_mx, pnl_my;
PNL_EXTERN Coord pnl_x, pnl_y;  /* screen coords if outside a panel */
                        /* where the mouse is */

PNL_EXTERN long int pnl_frame_number    PNL_INIT(0);
                        /* frames since last delay message */

PNL_EXTERN int pnl_delay        PNL_INIT(0);
                        /* frames till next message allowed from the queue */

PNL_EXTERN Boolean pnl_delayvirgin      PNL_INIT(TRUE);
                        /* flag prevents writing initial delay token */

PNL_EXTERN Boolean pnl_readscript       PNL_INIT(FALSE);
```

```
                    /* true when reading a script */

PNL_EXTERN Boolean pnl_writescript      PNL_INIT(FALSE);
                    /* true when writing a script */

PNL_EXTERN int pnl_scriptinfd           PNL_INIT(0);

PNL_EXTERN int pnl_scriptoutfd          PNL_INIT(0);

PNL_EXTERN char *pnl_scriptinfile       PNL_INIT("panel.script");

PNL_EXTERN char *pnl_scriptoutfile      PNL_INIT("panel.script");

PNL_EXTERN Boolean pnl_virgin           PNL_INIT(TRUE);
                    /* haven't done a dopanel yet */

PNL_EXTERN Boolean pnl_saveuserredraw   PNL_INIT(FALSE);
                    /* if the user is using them, requeue them */

PNL_EXTERN Screencoord pnl_ox, pnl_oy;
                    /* last window origin */

PNL_EXTERN Boolean pnl_justdown;
                    /* true when button first goes down */

PNL_EXTERN Boolean pnl_justup;
                    /* true when button first goes up */

PNL_EXTERN Boolean pnl_mousedown;
                    /* true when button is (was) down */
                    /* (according to the queue) */

PNL_EXTERN Boolean pnl_shiftkey         PNL_INIT(FALSE);
                    /* true when the shift key is down */

PNL_EXTERN Boolean pnl_controlkey       PNL_INIT(FALSE);
                    /* true when the control key is down */

PNL_EXTERN int pnl_winsave;
                    /* qid of (user's) window when dopanel() is called */

PNL_EXTERN int pnl_action_source        PNL_INIT(PNL_SRC_QUEUE);
                    /* where last mouse action came from */

PNL_EXTERN float pnl_char_threshold     PNL_INIT(PNL_CHAR_THRESHOLD);
                    /* see PNL_CHAR_THRESHOLD above */

PNL_EXTERN Boolean pnl_dont_draw        PNL_INIT(FALSE);
                    /* don't update panels (to avoid swapbuffers) */

PNL_EXTERN Boolean pnl_beveled          PNL_INIT(TRUE);
                    /* global control for disabling drawing bevels */

PNL_EXTERN int pnl_funcmode             PNL_INIT(PNL_FCNM_NONE);
                    /* role of called up-, active-, or downfuncs */

PNL_EXTERN Boolean pnl_ignore_delay     PNL_INIT(FALSE);
                    /* ignore delay packets when reading script */

PNL_EXTERN Boolean pnl_panel_bell       PNL_INIT(TRUE);
                    /* ring bell for hits on unselectable panels */

PNL_EXTERN int pnl_dopanel_return_mode  PNL_INIT(PNL_DRM_RETURN_PNL_CA);
                    /* to ground out dopanel() when in build mode */
```

```
#define PNL_FADE_PATTERN_INDEX          1
#define PNL_FADE_PATTERN_SIZE           16
#define PNL_FADE_PATTERN                                                \
/* wide diagonals */            { 0xf0f0, 0x7878, 0x3c3c, 0x1e1e,       \
                                  0x0f0f, 0x8787, 0xc3c3, 0xe1e1,       \
                                  0xf0f0, 0x7878, 0x3c3c, 0x1e1e,       \
                                  0x0f0f, 0x8787, 0xc3c3, 0xe1e1 }
#ifdef 0
/* fine diagonals */            { 0x9999, 0xcccc, 0x6666, 0x3333,       \
                                  0x9999, 0xcccc, 0x6666, 0x3333,       \
                                  0x9999, 0xcccc, 0x6666, 0x3333,       \
                                  0x9999, 0xcccc, 0x6666, 0x3333 }

                                { 0xf0f0, 0xf0f0, 0xf0f0, 0xf0f0,       \
                                  0x0f0f, 0x0f0f, 0x0f0f, 0x0f0f,       \
                                  0xf0f0, 0xf0f0, 0xf0f0, 0xf0f0,       \
                                  0x0f0f, 0x0f0f, 0x0f0f, 0x0f0f }

                                { 0x5555, 0x5555, 0x5555, 0x5555,       \
                                  0x5555, 0x5555, 0x5555, 0x5555,       \
                                  0x5555, 0x5555, 0x5555, 0x5555,       \
                                  0x5555, 0x5555, 0x5555, 0x5555 }
#endif

PNL_EXTERN short pnl_fade_pattern_index PNL_INIT(PNL_FADE_PATTERN_INDEX);
PNL_EXTERN short pnl_fade_pattern_size  PNL_INIT(PNL_FADE_PATTERN_SIZE);
PNL_EXTERN short pnl_fade_pattern[]     PNL_INIT(PNL_FADE_PATTERN);
                    /* pattern used to cover unselectable actuators */

/* function declarations */

/* panel library utilities */

#ifdef IRIS_4D
void *alloc(int);
void *pnl_alloc(int);
#else  IRIS_4D
void *alloc();
void *pnl_alloc();
#endif IRIS_4D

/* user functions */

PNL_EXTERN Panel    *mkpanel();
PNL_EXTERN Actuator *dopanel();
PNL_EXTERN void     dumpstate();
PNL_EXTERN void     drawpanel();
PNL_EXTERN short    userredraw();
PNL_EXTERN void     needredraw();
PNL_EXTERN Boolean  dumppanel();

PNL_EXTERN Panel    *pnl_mkpanel();
PNL_EXTERN Actuator *pnl_dopanel();
PNL_EXTERN void     pnl_dumpstate();
PNL_EXTERN void     pnl_drawpanel();
PNL_EXTERN short    pnl_userredraw();
PNL_EXTERN void     pnl_needredraw();
PNL_EXTERN Boolean  pnl_dumppanel();

#ifdef IRIS_4D

PNL_EXTERN Actuator *mkact(/*void (*)()*/); /* parameter decl doesn't seem to work */
PNL_EXTERN Actuator *_mkuseract(/*int, void (*)()*/); /* parameter decl doesn't seem to w
```

```
rk */
NL_EXTERN void newvalact(/*Actuator *,Panel *,Coord, Coord*/);
NL_EXTERN void addact(Actuator *,Panel *);
NL_EXTERN void addsubact(Actuator *,Actuator *);
NL_EXTERN void delact(Actuator *);
NL_EXTERN void endgroup(Panel *);
NL_EXTERN void addtogroup(Actuator *,Panel *);
NL_EXTERN void fixpanel(Panel *);
NL_EXTERN void fixact(Actuator *);
NL_EXTERN void labeloffsets(Actuator *);
NL_EXTERN void labeldimensions(Actuator *);
NL_EXTERN char *g_gets(Colorindex,Colorindex,Colorindex);
NL_EXTERN char *g_getstring(Colorindex,Colorindex,Colorindex,char *,int);
NL_EXTERN Boolean pnl_beginreadscript(char *);
NL_EXTERN Boolean pnl_beginwritescript(char *);
NL_EXTERN Boolean pnl_beginappendscript(char *);
NL_EXTERN void pnl_endreadscript();
NL_EXTERN void pnl_endwritescript();
NL_EXTERN float pnl_strwidth(Panel *, char *);
NL_EXTERN void pnl_listadd(Alist *, Alist **);
NL_EXTERN void pnl_listdelete(Alist *, Alist **);
NL_EXTERN Boolean pnl_listin(Alist *, Alist *);

NL_EXTERN Actuator *pnl_mkact(/*void (*)()*/); /* parameter decl doesn't seem to work *

NL_EXTERN Actuator *pnl__mkuseract(/*int, void (*)()*/); /* parameter decl doesn't seem
to work */
NL_EXTERN void pnl_newvalact(/*Actuator *,Panel *,Coord, Coord*/);
NL_EXTERN void pnl_addact(Actuator *,Panel *);
NL_EXTERN void pnl_addsubact(Actuator *,Actuator *);
NL_EXTERN void pnl_delact(Actuator *);
NL_EXTERN void pnl_endgroup(Panel *);
NL_EXTERN void pnl_addtogroup(Actuator *,Panel *);
NL_EXTERN void pnl_fixpanel(Panel *);
NL_EXTERN void pnl_fixact(Actuator *);
NL_EXTERN void pnl_labeloffsets(Actuator *);
NL_EXTERN void pnl_labeldimensions(Actuator *);
NL_EXTERN char *g_gets(Colorindex,Colorindex,Colorindex);
NL_EXTERN char *g_getstring(Colorindex,Colorindex,Colorindex,char *,int);
NL_EXTERN Boolean pnl_beginreadscript(char *);
NL_EXTERN Boolean pnl_beginwritescript(char *);
NL_EXTERN Boolean pnl_beginappendscript(char *);
NL_EXTERN void pnl_endreadscript();
NL_EXTERN void pnl_endwritescript();
NL_EXTERN float pnl_strwidth(Panel *, char *);
NL_EXTERN void pnl_listadd(Alist *, Alist **);
NL_EXTERN void pnl_listdelete(Alist *, Alist **);
NL_EXTERN Boolean pnl_listin(Alist *, Alist *);

.se  IRIS_4D

NL_EXTERN Actuator *mkact();
NL_EXTERN Actuator *_mkuseract();
NL_EXTERN void newvalact();
NL_EXTERN void addact();
NL_EXTERN void addsubact();
NL_EXTERN void delact();
NL_EXTERN void endgroup();
NL_EXTERN void addtogroup();
NL_EXTERN void fixpanel();
NL_EXTERN void fixact();
NL_EXTERN void labeloffsets();
NL_EXTERN void labeldimensions();
NL_EXTERN char *g_gets();
NL_EXTERN char *g_getstring();
NL_EXTERN Boolean pnl_beginreadscript();
NL_EXTERN Boolean pnl_beginwritescript();
NL_EXTERN void pnl_endreadscript();
NL_EXTERN void pnl_endwritescript();
NL_EXTERN Boolean pnl_beginappendscript();
NL_EXTERN float pnl_strwidth();
NL_EXTERN void pnl_listadd();
NL_EXTERN void pnl_listdelete();
NL_EXTERN Boolean pnl_listin();

PNL_EXTERN Actuator *pnl_mkact();
PNL_EXTERN Actuator *pnl__mkuseract();
PNL_EXTERN void pnl_newvalact();
PNL_EXTERN void pnl_addact();
PNL_EXTERN void pnl_addsubact();
PNL_EXTERN void pnl_delact();
PNL_EXTERN void pnl_endgroup();
PNL_EXTERN void pnl_addtogroup();
PNL_EXTERN void pnl_fixpanel();
PNL_EXTERN void pnl_fixact();
PNL_EXTERN void pnl_labeloffsets();
PNL_EXTERN void pnl_labeldimensions();
PNL_EXTERN char *g_gets();
PNL_EXTERN char *g_getstring();
PNL_EXTERN Boolean pnl_beginreadscript();
PNL_EXTERN Boolean pnl_beginwritescript();
PNL_EXTERN void pnl_endreadscript();
PNL_EXTERN void pnl_endwritescript();
PNL_EXTERN Boolean pnl_beginappendscript();
PNL_EXTERN float pnl_strwidth();
PNL_EXTERN void pnl_listadd();
PNL_EXTERN void pnl_listdelete();
PNL_EXTERN Boolean pnl_listin();

#endif IRIS_4D

/* initilization functions */

#ifdef IRIS_4D
extern void pnl_slider(Actuator *);
extern void pnl_vslider(Actuator *);
extern void pnl_hslider(Actuator *);
extern void pnl_dvslider(Actuator *);
extern void pnl_dhslider(Actuator *);
extern void pnl_filled_slider(Actuator *);
extern void pnl_filled_vslider(Actuator *);
extern void pnl_filled_hslider(Actuator *);
extern void pnl_button(Actuator *);
extern void pnl_wide_button(Actuator *);
extern void pnl_toggle_button(Actuator *);
extern void pnl_radio_button(Actuator *);
extern void pnl_left_arrow_button(Actuator *);
extern void pnl_right_arrow_button(Actuator *);
extern void pnl_up_arrow_button(Actuator *);
extern void pnl_down_arrow_button(Actuator *);
extern void pnl_left_double_arrow_button(Actuator *);
extern void pnl_right_double_arrow_button(Actuator *);
extern void pnl_up_double_arrow_button(Actuator *);
extern void pnl_down_double_arrow_button(Actuator *);
extern void pnl_meter(Actuator *);
extern void pnl_analog_meter(Actuator *);
extern void pnl_analog_bar(Actuator *);
extern void pnl_strip_chart(Actuator *);
```

```
xtern void pnl_scale_chart(Actuator *);
xtern void pnl_puck(Actuator *);
xtern void pnl_floating_puck(Actuator *);
xtern void pnl_rubber_puck(Actuator *);
xtern void pnl_typein(Actuator *);
xtern void pnl_typeout(Actuator *);
xtern void pnl_label(Actuator *);
xtern void pnl_mouse(Actuator *);
xtern void pnl_slideroid(Actuator *);
xtern void pnl_palette(Actuator *);
xtern void pnl_vpalette(Actuator *);
xtern void pnl_hpalette(Actuator *);
xtern void pnl_dial(Actuator *);
xtern void pnl_multislider(Actuator *);
xtern void pnl_hmultislider(Actuator *);
xtern void pnl_vmultislider(Actuator *);
xtern void pnl_multislider_bar(Actuator *);
xtern void pnl_multislider_open_bar(Actuator *);
xtern void pnl_vmultislider_bar(Actuator *);
xtern void pnl_vmultislider_open_bar(Actuator *);
xtern void pnl_hmultislider_bar(Actuator *);
xtern void pnl_hmultislider_open_bar(Actuator *);
xtern void pnl_menu(Actuator *);
xtern void pnl_icon_menu(Actuator *);
xtern void pnl_sub_menu(Actuator *);
xtern void pnl_menu_item(Actuator *);
xtern void pnl_icon(Actuator *);
xtern void pnl_frame(Actuator *);
xtern void pnl_cycle(Actuator *);
xtern void pnl_scroll(Actuator *);
else  IRIS_4D
xtern void pnl_slider();
xtern void pnl_vslider();
xtern void pnl_hslider();
xtern void pnl_dvslider();
xtern void pnl_dhslider();
xtern void pnl_filled_slider();
xtern void pnl_filled_vslider();
xtern void pnl_filled_hslider();
xtern void pnl_button();
xtern void pnl_wide_button();
xtern void pnl_toggle_button();
xtern void pnl_radio_button();
xtern void pnl_left_arrow_button();
xtern void pnl_right_arrow_button();
xtern void pnl_up_arrow_button();
xtern void pnl_down_arrow_button();
 tern void pnl_left_double_arrow_button();
 tern void pnl_right_double_arrow_button();
 tern void pnl_up_double_arrow_button();
 tern void pnl_down_double_arrow_button();
 tern void pnl_meter();
 tern void pnl_analog_meter();
 tern void pnl_analog_bar();
 tern void pnl_strip_chart();
 tern void pnl_scale_chart();
 tern void pnl_puck();
 tern void pnl_floating_puck();
 tern void pnl_rubber_puck();
 tern void pnl_typein();
 tern void pnl_typeout();
 tern void pnl_label();
 tern void pnl_mouse();
 tern void pnl_slideroid();
```

```
extern void pnl_palette();
extern void pnl_vpalette();
extern void pnl_hpalette();
extern void pnl_dial();
extern void pnl_multislider();
extern void pnl_vmultislider();
extern void pnl_hmultislider();
extern void pnl_multislider_bar();
extern void pnl_multislider_open_bar();
extern void pnl_vmultislider_bar();
extern void pnl_vmultislider_open_bar();
extern void pnl_hmultislider_bar();
extern void pnl_hmultislider_open_bar();
extern void pnl_menu();
extern void pnl_icon_menu();
extern void pnl_sub_menu();
extern void pnl_menu_item();
extern void pnl_frame();
extern void pnl_cycle();
extern void pnl_scroll();
#endif IRIS_4D

#endif PNL_EDITOR_PARSING
```