GEO - an application software for the presentation of zircon morphometry data

Dušan Dugáček [1], Ján Hronec [2], Gejza M. Timčák [1] a Eva Hroncová [1]

Geo - aplikačný program pre prezentáciu morfometrických dát zirkónu
Zirkóny sú jedným z najzaujímavejších indikačných minerálov. Vyskytujú sa vo vulkanických, plutonických, sedimentárnych aj metamofovaných horninách. V r.1976 J.P. Pupin vyvinul systém interpretácie dát o morfometrii zirkónov, ktorý sa užíva pri výskume procesov zmien podmienok kryštalizácie či premien hornín. Morfometrické a mikrochemické údaje o zirkónoch boli uložené do databáz geografického informačného systému (GIS) v rámci GIS projektu "Zirkón", pre zabezpečenie efektívnej práce na regionálných a nadregionálnych koreláciách. GIS bol realizovaný v prostredí *Intergraph MicroStation PC* (verzia 5). Príspevok popisuje jeden z aplikačných programov pre toto prostredie - GEO, ktorý umožňuje počítačovú prezentáciu a interpretáciu rozloženia morfometrických priemerov zirkónových vzoriek. Polia morfometrických priemerov sú kontúrovateľné a sú na pozadí vektorovej šablóny Pupinovho typogramu s vyznačenými typologickými evolučnými trendmi.

Key words: Pupin typogram, zircon morphometry, zircon microchemistry, evolution trend, database, GIS.

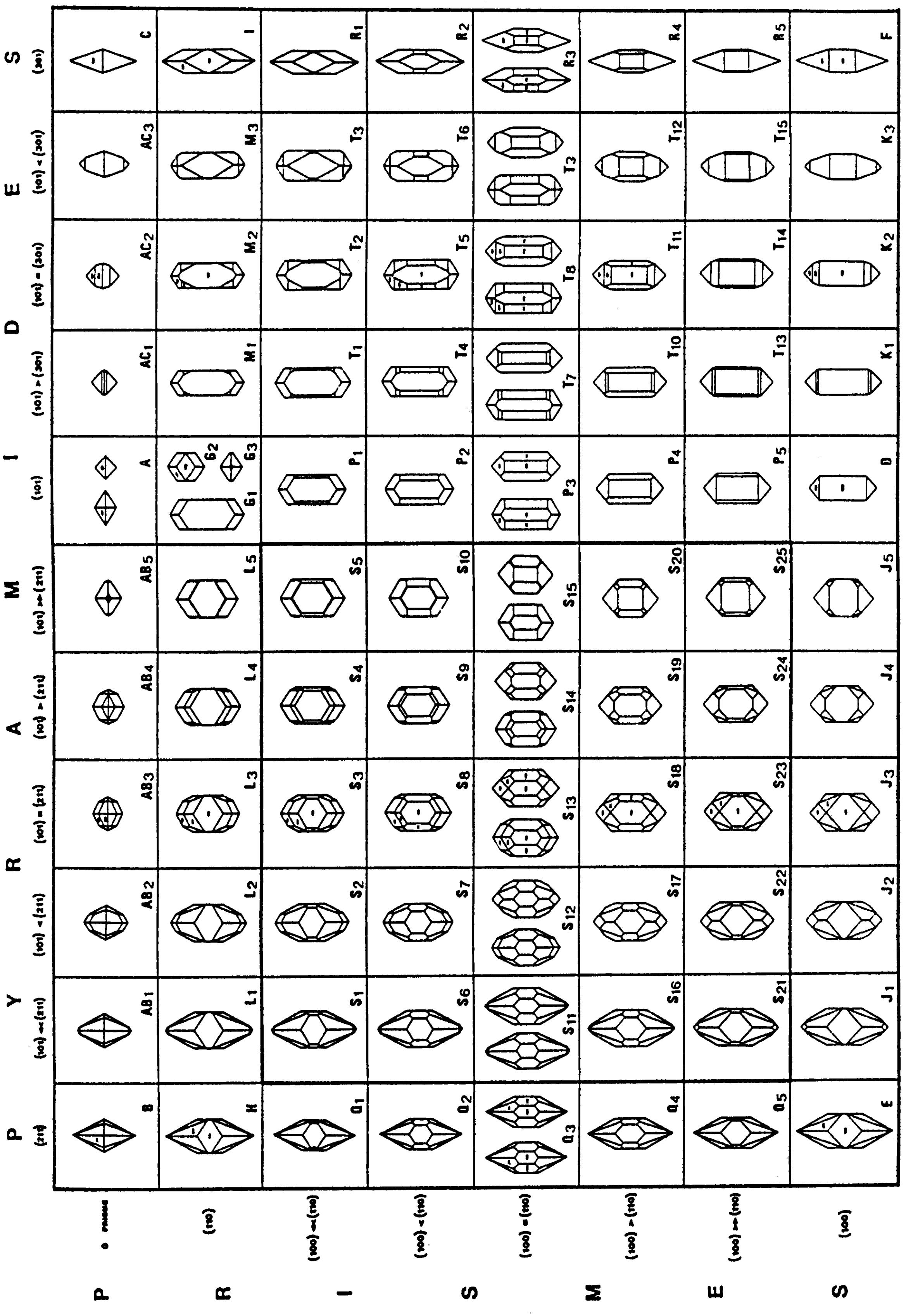
Introduction

Zircon is a mineral that can be found in magmatic, sedimentary and metamorphic rocks alike. J.P. Pupin (1976) defined the quantitative relationship between the development of zircon crystallographic surfaces and their crystallisation history. Later, he also defined how the microchemical properties of zircons can reveal the character of magmatic phase developments (Pupin, 1992). Zircons in various rocks from Europe, S. America and Nepal were successfully used in the classification
of granites, the genetic classification of rhyolites, the study of the role of fluid phase and specially
of water in the process of the crystallisation of the melt, the study of the relationship between plutonism and volcanism, the study of the origin of enclaves in magmatic rocks, the study of the character
of the protolith of anatectic rocks, the study of magmatic zoning and in a number of other areas (ibid.).

Since 1979, in the Department of Geology and Mineralogy of the BERG Faculty
of the Technical University of Košice, we have worked with the morphometry of zircons from Neogene and Paleogene rocks of the Western Carpathians. As soon as powerful GIS systems
(e.g. the Intergraph MicroStation PC & MGE PC) became available to us in 1990, we started building our geologically - oriented GIS. We prepared geographical and geological digital (vector type) maps (1 : 200 000) with sampling loci linked to alphanumeric and graphical datafiles and databases. Presently - as the official 1:50 000 and 1:10 000 scale digital maps became available - we are transferring the geological information layers and our data on zircons into these maps. A simple software is being developed for "seamless" map sheet handling. The alphanumerical and graphical databases are managed by the DBMAN application program (Dugáček and Timčák, 1995a), if
the data structure varies within the sets of sample points, the alphanumerical datafiles and graphical databases are managed by the IMAGER programme (Dugáček and Timčák, 1995b). The IMAGER software was used mainly to work with zircon data because in this way each data record linked with
a sampling point can have its own structure. This was necessary in order to cope with the fact that data The earliest data types represent random point analyses of random zircon grains from a given locality. The latest ones are series of point analyses of strategically selected loci on zircon grain sections of preselected morphometric types. For data synthesis and presentation we needed

[1] Katedra geológie a mineralógie F BERG Technickej univerzity, 043 84 Košice, Park Komenského 15
[2] MicroCAD Systems, 040 01 Košice, Letná 45
(Recenzovali: RNDr. István Tarján a Doc.RNDr. Anton Laurin, CSc. Revidovaná verzia doručená 22.4.1997)

magmatic evolution trend templates that would facilitate research as well as education, and which would enable us to work with large specimen populations.

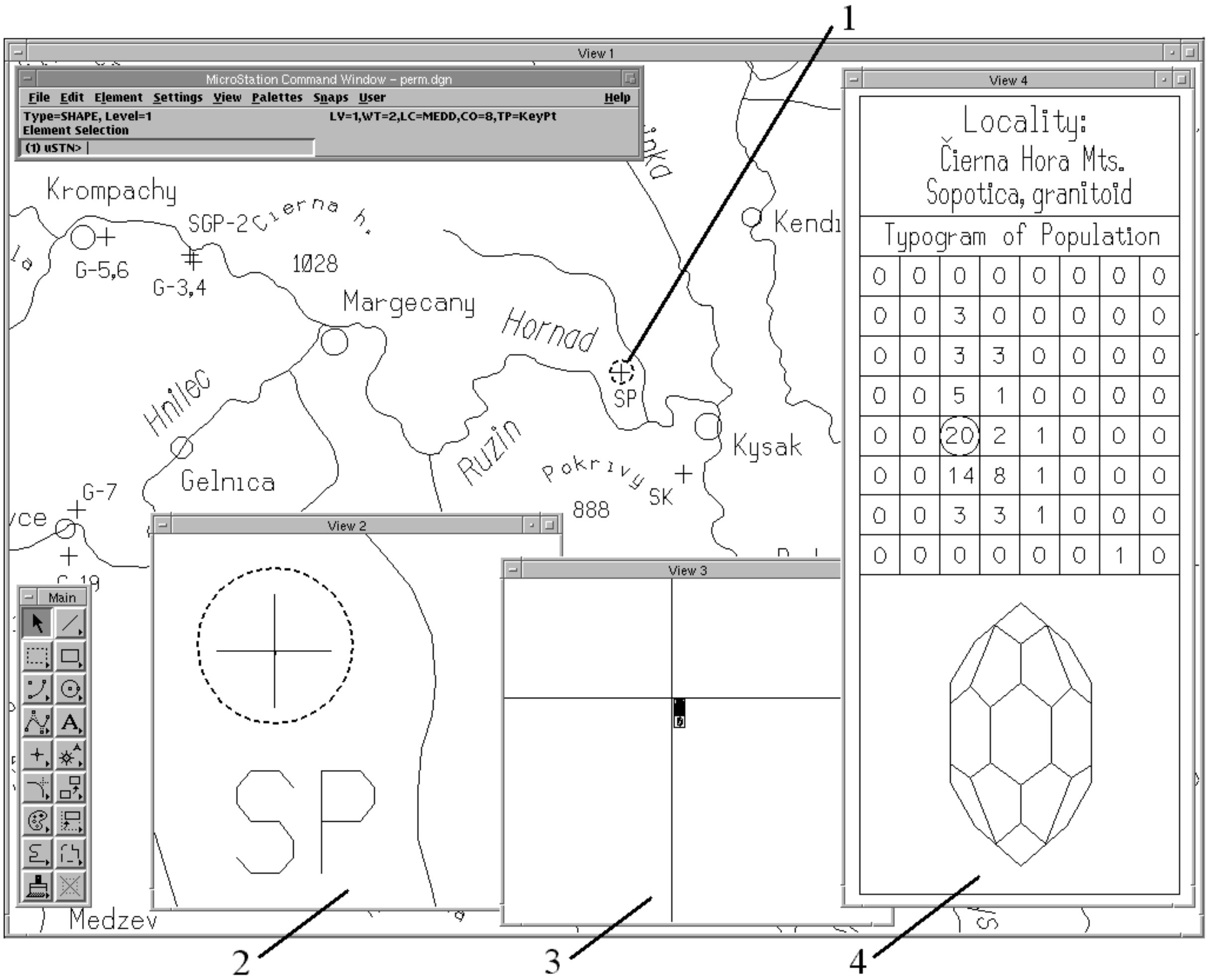


P Y R A M I D E S
P R I S M E S
C I R1 R2 R3 R4 R5 F
AC3 M3 T3 T6 T9 T12 T15 K3
AC2 M2 T2 T5 T8 T11 T14 K2
AC1 M1 T1 T4 T7 T10 T13 K1
A G1 G2 G3 P1 P2 P3 P4 P5 D
AB5 L5 S5 S10 S15 S20 S25 J5
AB4 L4 S4 S9 S14 S19 S24 J4
AB3 L3 S3 S8 S13 S18 S23 J3
AB2 L2 S2 S7 S12 S17 S22 J2
AB1 L1 S1 S6 S11 S16 S21 J1
B H Q1 Q2 Q3 Q4 Q5 E

Fig.1a. The extended Pupin typogram showing the relative development of pyramidal and prismatic surfaces of zircon grains used for the typological classification.

The Pupin classification

In Pupin´s classification system (Pupin, 1976), the evolution of the melt from which a rock crystallised can be characterised in the most robust way when a studied rock specimen is represented by its typological mean point. The typological mean points are calculated from the data in the basic 8x8 typological matrix. The matrix contains a row- and column-wise consistent array of zircon crystal face development types. The zircons - after being separated from the host-rock and specially mounted, are observed and classified under the polarising microscope. A minimum number of 100 grains per specimens are to be analysed for the typological assessment. The frequency data of the occurrence of zircons of various typological developments are entered into the typological matrix. Thus a 2D matrix is formed. Figure 1a shows the extended (11x8) typological matrix template.



MicroStation Command Window – perm.dgn

```
Type=SHAPE, Level=1                    LV=1,WT=2,LC=MEDD,CO=8,TP=KeyPt
Element Selection
(1) uSTN> |
```

Locality:
Čierna Hora Mts.
Sopotica, granitoid

Typogram of Population

0	0	0	0	0	0	0	0
0	0	3	0	0	0	0	0
0	0	3	3	0	0	0	0
0	0	5	1	0	0	0	0
0	0	20	2	1	0	0	0
0	0	14	8	1	0	0	0
0	0	3	3	1	0	0	0
0	0	0	0	0	0	1	0

Fig.1.b. A zircon typogram zoomed in (2)-(4) from a sample location point (1) on the digital map. The zircon crystal of S12 type, shown under the typogram (4) represents the most frequently occurring type within the given locality.

The columns for types with surfaces (101>301), (101=301) and (101<301) are not used in the basic typogram (Pupin, 1980) as these types were not observed yet under natural conditions. The numbers in brackets denote the crystallographic surfaces of zircon grains and their relative development, which serve as the classification criteria of the observed zircon grains. Figure 1.b. shows a typogram attached to a sample point in the Intergraph environment. The co-ordinates of the mean point of a specimen are calculated as the weighted arithmetic mean of the X- and Y-wise projected distribution frequency data in the matrix. The projected frequency data are obtained in such a way that the numbers of grains of the given grain population in the individual rows and subsequently of individual columns, are added. The resulting column and row of sums form the projected distribution frequency data. The formula needed to compute that distribution is given e.g. in Pupin (1976). The position of the mean point in the typogram serves as the basis of the geological interpretative work that uses zircon typology data. For magmatic rocks, the position of mean points of the analysed specimens can be used to deduce the evolution trend of magma from which the rocks in question originated. The template for the evaluation of the magmatic evolution trend was described by Pupin (1976, 1980). It enables the separation of "primitive" from evolved magmas on the basis of zircon typological data. These data are often complemented by microprobe point analyses (U, Th, Y, Hf) of zircon grain sections (Pupin, 1992).

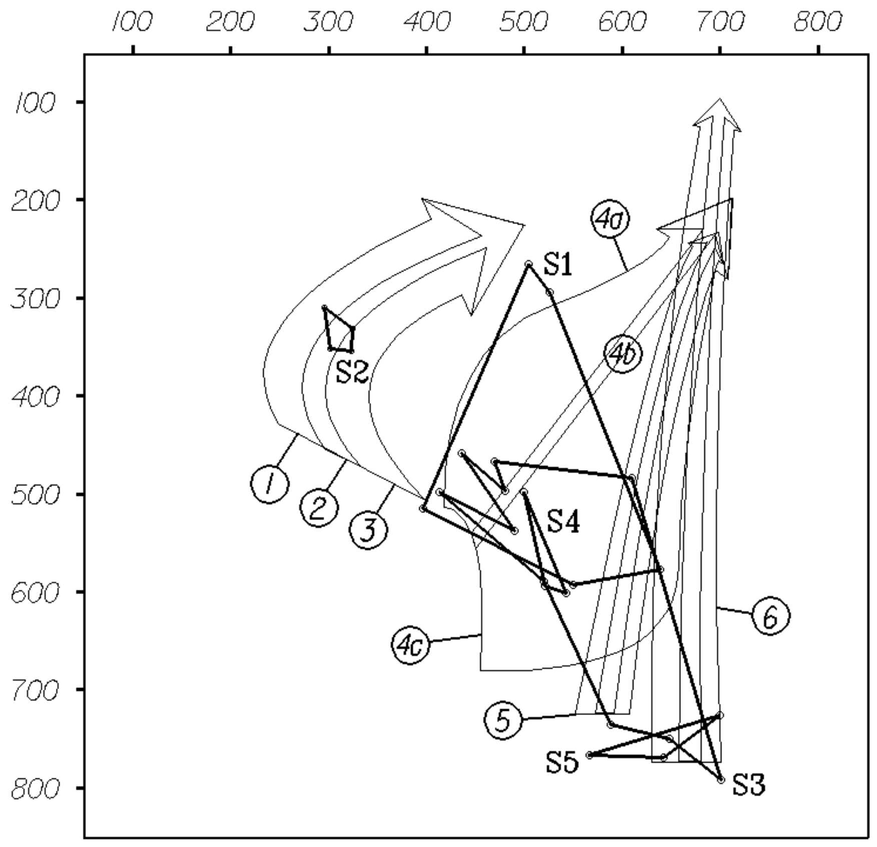


Fig.2. The plot of the typological mean points for zircon populations from Paleozoic specimens from the West Carpathians, using the GEO application programme. Set 1 (S1) represents specimens from N. Tatricum, set 2 (S2) specimens from N. Veporicum, set 3 (S3) specimens from N. Gemericum, set 4 (S4) specimens from S. Veporicum, set 5 (S5) specimens from the Zemplinicum unit.

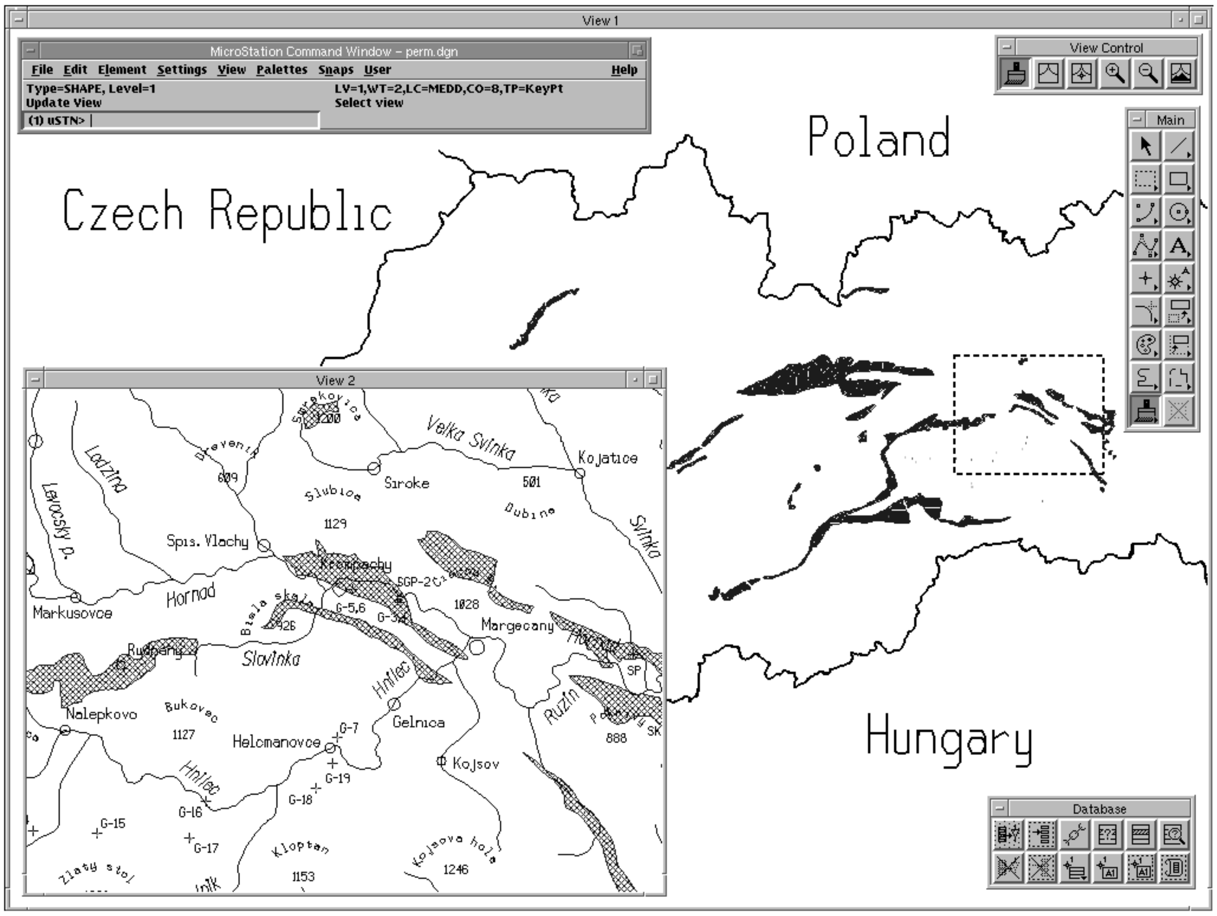
Numbers (1)-(6) denote the evolution trend fields. The left three-in-one arrow in the template indicates aluminous leucogranites (1), (sub) autochtonous monzogranites & granodiorites (2) and intrusive aluminous monzogranites & granodiorites (3). The middle three-in-one arrow indicates granites of the ca-alk series (4a-c). The next arrow (5) granites of the subalcaline series and the last (6) granites of the alkaline series. The scale of values of the X-wise (IA) and Y-wise (IT) coordinates of the typological mean points are 100-800. These values follow the convention for ascribing numerical values to typological fields (cf. Fig.1) set by J.P. Pupin.

The GEO application

In this paper it is not possible to give a full account of how the MicroStation graphical environment works, and we have to make reference to the appropriate Intergraph publication (Intergraph, 1990). The GEO application program works in this environment and enables the graphical presentation of the morphometric mean points of zircon populations in the different subregions within the Pupin magmatic evolution trend diagram. The data required for plotting are selected from dBase IV database, which is managed through the DBMAN application.

An example of application

As an example of the application of GEO, Figure 2 shows populations of typological mean points of Paleozoic rocks from different units of the Western Carpathians (data taken from: Broska et al. 1993, Uher et al. 1995, Jablonská et al. 1995, cf. Fig.3). The mean points were calculated in the way described earlier. The program permits an easy change or extension of the displayed sample populations through pop-up menus. It also enables the identification of mean points of specimens coming from the various geological subregions or units - by a polygon distinguishable by colour or hatching. The magmatic evolution trend template facilitates a rapid assessment of the evolution of the melt from which the zircons have crystallised. In Figure 2. it is shown that the least evolved magma was that of the specimen group S5; groups S2 and S1 crystallised from the most evolved magmas.



```
Type=SHAPE, Level=1                    LV=1,WT=2,LC=MEDD,CO=8,TP=KeyPt
Update View                            Select view
(1) uSTN> |
```

Fig.3. The Permian of West Caprathians (View 1) and the sample points shown on the zoomed digital map (View 2). View 2 shows the locations of sample points in the East Slovakian region (crosses with alphanumerical codes). The rectangle in the map shows the location of the enlarged insert. The pop-up menu in the right lower corner enables the work with the zircon database.

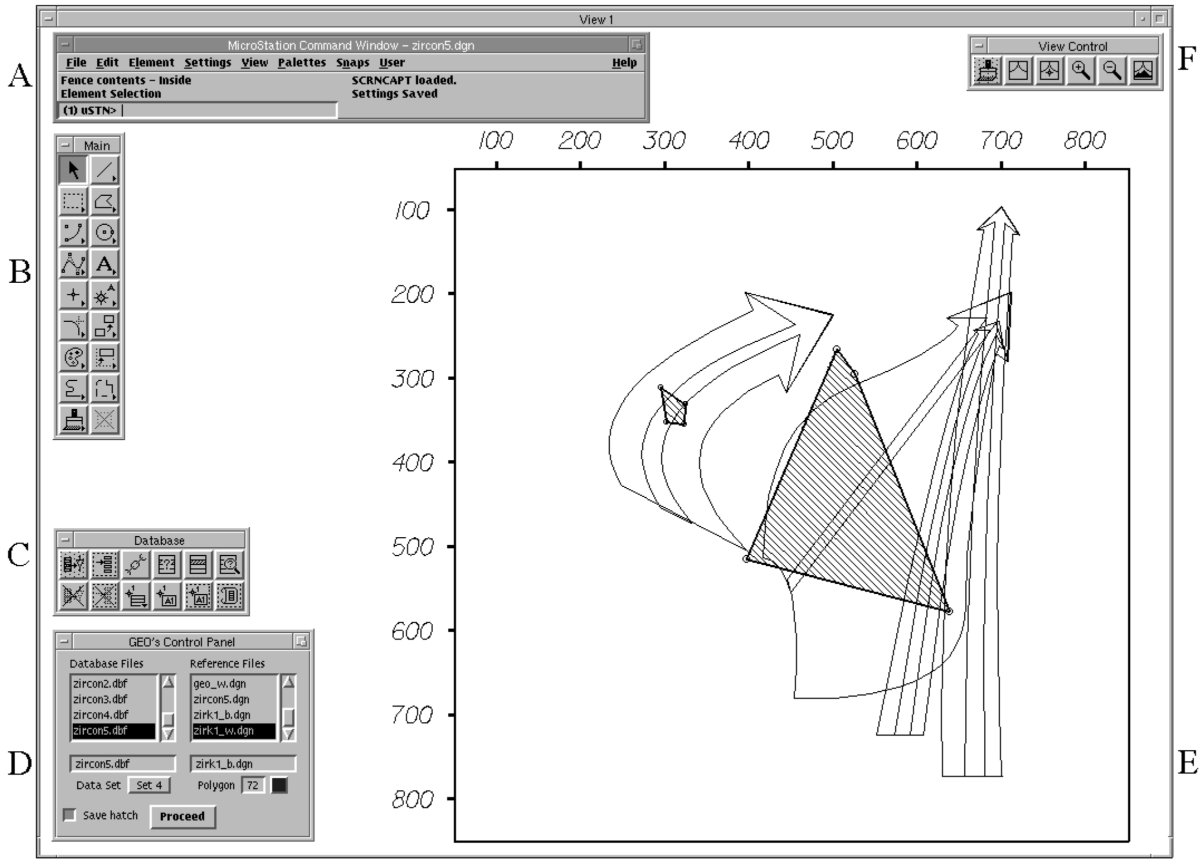
The GEO program

The GEO program was written in MicroStation Development Language (MDL). It can be run on a Pentium type PC under Microsoft Windows NT or under MS DOS. The MDL language is an ANSI standard C language implementation, that in addition to standard C features, includes more than 800 specialised procedures and functions. For the user, the application programs can thus create an environment that compares with the comfort of Windows™ and is fully consistent with the MicroStation environment.

The PC has to have memory large enough for accommodating the MicroStation (version 5). Practically it means at least 16MB of RAM and a fast hard disk.

The GEO is compiled and linked from program modules given below. First, the geo.mc (base resource file), geo.r (resource source file), geo.h (header file), geocmd.r (command table source file), geotyp.mt (type of variables file) and the geo.mke (the Make file) had to be written, The geo.mc is the core of the application, the geo.r contains the definition of item resources used in the application, geo.h contains definitions shared by multiple source files (like the geo.mc, geo.r

and geocmd.h), The geocmd.r contains definitions for user implemented commands, geotyp.mt is a "type file" for the generation of definitions used in MDL C-expression built-in functions and finally the geo.mke is a "make file" in which it is stated how to build one's application. It is executed by the bmake.exe utility. The bmake.exe calls step by step the compilers and linker executivs.
The source text of the geo.mc is given in Annex 1. By the compilation and linking, the following temporary modules are formed: geo.rsc (resource file from geo.r), geocmd.rsc, geotyp.rsc, geotyp.r, geocmd.h, geo.mo (object file from geo.mc) and geo.mp (simplified form of the application created by merging object filed with MDL library functions). The final module is the geo.ma, which is then called by the user in the MicroStation environment.



Fence contents - Inside
Element Selection
SCRNCAPT loaded.
Settings Saved
(1) uSTN>

Database Files: zircon2.dbf, zircon3.dbf, zircon4.dbf, zircon5.dbf
Reference Files: geo_w.dgn, zircon5.dgn, zirk1_b.dgn, zirk1_w.dgn
zircon5.dbf
zirk1_b.dgn
Data Set Set 4
Polygon 72
Save hatch
Proceed

Fig.4. Screen-capture view of the Pupin diagram template. Items A-D are described in the text. E is the template of the Pupin typological evolution trends with sample groups from different geological subregions surrounded by two polygons. One of the evolution trend arrows ((5) of Fig.2) is switched off for better clarity of view.

Figure 4 shows a view into the MicroStation CAD environment. Object A is a MicroStation Command Window (MCW) which is a dialogue box. B is the main tool palette of the graphical functions. This tool palette makes the utilities for hand drawing accessible. The MCW dialogue box enables access to all system functions through pull-down menus. Thus from the File menu, using the New... function, it is possible to open a new design file, or - using the Save as ... function to save an image into a file. On the upper part of the MCW dialog box the name of the active design file can be seen (bb.dgn), which contains the diagram. This design file contains graphical information, but it is possible to link alphanumerical data with its graphic elements. This information can be accessed through the Database dialog box C. This box contains a tool palette, which can be opened through the MCW. The alphanumeric data are stored in the tables within databases of xBase type.
On the Database dialog box, the Review Database Attributes function button (upper right corner) is activated. It enables the review of the content of files attached to graphic elements in the map.

The GEO Control Panel (GCP) is marked with D. It is the dialog box of the GEO application. The work with this application is as follows:

First it is necessary to prepare a database table, which contains the input for processing by the application. On the left hand side of the GCP (D) the Database Files list box can be seen.
The database tables are in the files, from which one can select the one that contains the required data. These can be further processed by the GEO application. The selection is done by clicking on

some
of the file items with the mouse button. After this, the selected text item appears in the text item box situated under the list box. In the given example the xyr.dbf was selected. The Data Set switch
is an option button menu, which enables the user to select that file group from the given table, which have identical Set attribute value. The polygons in the diagrams are traced by such batch processing. On the right hand side of the GCP, the Reference Files list box can be seen. These files were created in the MicroStation environment (they have a "design file" - .dgn extension). They contain the empty template of the evolution trend diagram. The user can opt for different varieties of the template by clicking on an item in the list box. The name of the selected file will appear in the text item box under the list box. In our example the po_2.dgn file was selected. Another user tool is the Polygon colour picker. It enables the access to 256 colours displayed in a chess-board pattern. The selected colour will dye the polygon. In our case colour No.72 was selected.

When all the parameters have been set, clicking on the Proceed button displays a new polygon (a new group of mean points). At the outset, the bb.dgn design file was empty. Upon the appearance of the template with the first polygon, it is stored in this file. Every successive activation of the Proceed function, enables the processing of a new point set and to create a new surrounding polygon.
The nodes of the polygons are graphic objects, to which access to database tables is linked. This enables - through the use of the Database dialog box - the tracking of alphanumeric information related to the image. The alphanumeric data are then displayed in a scrolling dialog box showing items in the database.

Conclusions

The paper shows the purpose, structure and use of an application program
for the presentation of basic morphometric data of zircons. By its use, it is possible to assess
and correlate large sets of typological data of zircon populations that can be used in geological research. Such approach was successfully used also in the investigation of Paleozoic magmatism
in the West Carpathians. The software and the results were successfully employed in graduate tuition, Summer Schools as well as Workshops related to GIS and zircon morphometry.

References

Broska, I., Vozár, J., Uher, P.& Jakabská K.: The typology of zircons from Permian rhyolites-dacites and their pyroclastics (W. Carpathians). *Proc. Conf.: The geodynamic model and the deep structure of the West Carpathians, D. Štúr Geological Institute, Bratislava, 1993, SK, In Slovak, pp. pp. 151-158.*

Dugáček, D.& Timčák, G.M.: The application of the DBMAN program for a regional ecological GIS. *In Slovak, GIS and DPZ Ostrava CZ, 1995a. 6 pp.*

Dugáček, D.& Timčák, G.M.: Application of the IMAGER program within the investigation of zircons from Variscan granitoides of the Čierna Hora Mts. *In Slovak, Geoforum, Bratislava SK, 2, 1995b, pp.34-36.*

Intergraph: MicroStation PC Reference Guide. *Intergraph Corp., Huntsville, Alabama, 1990.*

Jablonská, J., Pupin, J.P.& Timčák G.M.: Morphological and microchemical assessment of zircons
in granite specimens from the Čierna hora Mts. (Western Carpathians). *Geologica Carpathica, Bratislava, SK, 46,4, 1995, pp.241-251.*

Pupin, J.P.: Signification des caractéres morphologiques du zircon commun des roches en pétrologie. *Base de la méthode typologique. Thése Doct Sciences d'Etat, Université de Nice, 1976, 394pp.*

Pupin, J.P.: Zircon and granite petrology. *Contrib. Mineral. Petrol., 73, 1980, pp.207-220.*

Pupin, J.P.: Les zircons des granites océaniques et continentaux: couplage typologie - géochimie des éléments en traces. *Bull. Soc.Geol. France, T.163, No.4, 1992, pp.495-507.*

Timčák, G.M., Orlitová, E., Jablonská, J.& Jakabská, K.: Database of chemical, microchemical and morphometric data on zircons from West Carpathians. *Geologické práce, Správy,Bratislava, SK, 99, 1994, pp.109-112.*

Uher, P., Snopko, L., Vozár, J., Broska I.& Jablonská, J.: Typology of zircons from Paleozoic metavolcanites of the Gelnica unit of the Gemeric crystalline complex, Geologica Carpathica,
in print, 1995, 24pp.

Annex 1.: ***Listing of the geo.mc programme***

```
/*---------------------------------------------------------------------+
|  GEO - MDL Application                                 version 1.5             |
+---------------------------------------------------------------------*/
/*---------------------------------------------------------------------+
|  (C)  Dugacek, Hronec 1995-1997                                                  |
+---------------------------------------------------------------------*/
/*---------------------------------------------------------------------+
|  Include Files                                                                   |
+---------------------------------------------------------------------*/
#include    <mdl.h>                /* MDL Library funcs structures & constants        */
#include    <dlogitem.h>           /* Dialog Box structures & constants               */
#include    <cexpr.h>             /* C Expression structures & constants             */
#include    <cmdlist.h>           /* MicroStation command numbers                    */
#include    <dlogman.fdf>          /* dialog box manager function prototypes          */
#include    <keys.h>              /* MStation keyboard definitions                   */
#include    <msdefs.h>            /* MStation data structures                        */
#include    <global.h>           /* structure of global data areas in MS            */
#include    <mselems.h>           /* definitions of MS elements                      */
#include    <dbdefs.h>            /* Database structures & constants                 */
#include    <dberrs.h>           /* Database error codes explanations               */
#include    <filelist.h>         /* file definitions for MS file functions          */
#include    "geo.h"              /* geo dialog box example constants & structs      */
#include    "geocmd.h"          /* geo dialog box command numbers                  */


Private GeoGlobals       geoGlobals = {72, "", "", 1, 0};


void geo_dialogHook();
void geo_DBFHook();
void geo_REFHook();


Private DialogHookInfo uHooks[] =
    {
    {HOOKDIALOGID_Geo,        geo_dialogHook},
    {HOOKITEMID_ListDBF,      geo_DBFHook},
    {HOOKITEMID_ListREF,      geo_REFHook},
    };


/* ----- Main routine --------------------------------------------------------   --------------*/
Public main
(
int   argc,                     /* => number of args in next array */
char *argv[]                /* => array of cmd line arguments */
)
    {
    char          *setP;
    RscFileHandle        rscFileH;
    DialogBox     *dbP;

    mdlResource_openFile (&rscFileH, NULL, 0);


    if (mdlParse_loadCommandTable (NULL) == NULL)
           errorPrint (1);
    setP = mdlCExpression_initializeSet (VISIBILITY_DIALOG_BOX, 0, FALSE);
    mdlDialog_publishComplexVariable
                  (setP, "geoglobals","geoGlobals", &geoGlobals);
    mdlDialog_hookPublish
                  (sizeof(uHooks)/sizeof(DialogHookInfo), uHooks);


    /* open the "Geo" dialog box */
    if ( (dbP = mdlDialog_open (NULL, DIALOGID_Geo)) == NULL)
           errorPrint (2);
    }


/* -----------------------------------------------------------------+
|  Hook Functions                                                                |
+------------------------------------------------------------------*/
/* ----- GEO dialog box hook ----------------------------------------*/
Private void      geo_dialogHook
(
DialogMessage *dmP       /* => a ptr to a dialog message */
)
    {
    /* ignore any messages being sent to modal dialog hook */
    if (dmP->dialogId != DIALOGID_Geo)
           return;


    dmP->msgUnderstood = TRUE;
    switch (dmP->messageType)
           {
           case DIALOG_MESSAGE_DESTROY:
              {
              /* unload this mdl task when the Geo Dialog is closed */
              mdlDialog_cmdNumberQueue (FALSE, CMD_MDL_UNLOAD,
                                       mdlSystem_getCurrTaskID(), TRUE);
              break;
              };


           default:
              dmP->msgUnderstood = FALSE;
              break;
           }
    }


/* ----- List box hook (Database Files) ------------------------------*/
Private void geo_DBFHook
```

```
(
DialogItemMessage   *dimP       /* => a ptr to a dialog item message */
)
   {
   dimP->msgUnderstood = TRUE;


   switch (dimP->messageType)
          {
      case DITEM_MESSAGE_CREATE:
              {
         StringList   *fileListP;


         fileListP = mdlStringList_create(0,1);
         mdlFileList_get
                  (fileListP , FILELISTATTR_FILES, PATH_TO_DBF, "*.DBF");
         mdlDialog_listBoxSetStrListP
                  (dimP->dialogItemP->rawItemP, fileListP, 1);
            if (mdlStringList_size (fileListP) == 0)
                  {
                  char     msg[64], outMsg[64];


                  strcpy (msg, "Path to DBF: %s");
                  sprintf (outMsg, msg, PATH_TO_DBF);
                  mdlDialog_openAlert (outMsg);
                  }
            break;
            };
      case DITEM_MESSAGE_DESTROY:
              {
         StringList   *fileListP;


         fileListP = mdlDialog_listBoxGetStrListP
                                                     (dimP->dialogItemP->rawItemP);
         mdlStringList_destroy(fileListP);
            break;
            };
case DITEM_MESSAGE_BUTTON:
            {
         StringList   *fileListP;
         int         mr,xr,mc,xc;
         char        *strP;


         if (dimP->u.button.buttonNumber == DATAPNT &&
            dimP->u.button.buttonTrans  == BUTTONTRANS_UP &&
            dimP->u.button.upNumber     == 2)
           {
            mdlDialog_listBoxGetSelectRange
                           (&mr,&xr,&mc,&xc,dimP->dialogItemP->rawItemP);
            fileListP = mdlDialog_listBoxGetStrListP
                           (dimP->dialogItemP->rawItemP);
            mdlStringList_getMember(&strP,NULL,fileListP,mr);
            sprintf (geoGlobals.DBF, "%s",strP);
            mdlDialog_synonymsSynch(NULL,SYNONYMID_DBF,NULL);
           }
```

```
            break;
            };
      default:
            dimP->msgUnderstood = FALSE;
            break;
      }
   }


/* ----- List box hook (Reference Files) ------------------------------*/
Private void geo_REFHook
(
DialogItemMessage   *dimP       /* => a ptr to a dialog item message */
)
   {
   dimP->msgUnderstood = TRUE;


   switch (dimP->messageType)
          {
      case DITEM_MESSAGE_CREATE:
            {
         StringList   *fileListP;


         fileListP = mdlStringList_create(0,1);
         mdlFileList_get
                  (fileListP, FILELISTATTR_FILES, PATH_TO_DGN, "*.DGN");
         mdlDialog_listBoxSetStrListP
                  (dimP->dialogItemP->rawItemP, fileListP, 1);
            if (mdlStringList_size (fileListP) == 0)
                  {



                  strcpy (msg, "Path to DGN: %s");
                  sprintf (outMsg, msg, PATH_TO_DGN);
                  mdlDialog_openAlert (outMsg);
                  }
            break;
            };
      case DITEM_MESSAGE_DESTROY:
            {
         StringList   *fileListP;


         fileListP = mdlDialog_listBoxGetStrListP
                                             (dimP->dialogItemP->rawItemP);
         mdlStringList_destroy(fileListP);
            break;
            };
case DITEM_MESSAGE_BUTTON:
            {
         StringList   *fileListP;
         int          mr,xr,mc,xc;
         char         *strP;


         if (dimP->u.button.buttonNumber == DATAPNT &&
            dimP->u.button.buttonTrans  == BUTTONTRANS_UP &&
            dimP->u.button.upNumber     == 2)
           {
            mdlDialog_listBoxGetSelectRange
                                    (&mr,&xr,&mc,&xc,dimP->dialogItemP->rawItemP);
            fileListP = mdlDialog_listBoxGetStrListP
                                    (dimP->dialogItemP->rawItemP);
            mdlStringList_getMember(&strP,NULL,fileListP,mr);
            sprintf (geoGlobals.REF, "%s",strP);
            mdlDialog_synonymsSynch(NULL,SYNONYMID_REF,NULL);
           }
             break;
             };
         default:
             dimP->msgUnderstood = FALSE;
             break;
     }
  }


/*-------------------------------------------------------------------+
|      Command Handling routines                                      |
+-------------------------------------------------------------------*/
/* ----- Command routine for the PROCEED pushbutton -------------------*/
Public cmdName void geo_proceed
(
char      *unparsedP       /* => unparsed part of command */
)
cmdNumber   CMD_PROCEED
  {
  if (area())
         {
         errorPrint (3);
         return;
         }
  }


/*-------------------------------------------------------------------+
|  Utility routines                                                   |
+-------------------------------------------------------------------*/
Private void errorPrint
(
int errorNumber                /* => number of error to print */
)
  {
  char    errorMsg[80];


  if (mdlResource_loadFromStringList (errorMsg, NULL,
                    MESSAGELISTID_GeoErrors, errorNumber))
         return;  /* unable to find message with number "errorNumber" */


  mdlDialog_dmsgsPrint (errorMsg);
  }


/*-------------------------------------------------------------------+
|  Program routines                                                   |
+--------------------------------------------------------------------*/
/* ----- Prepare working area -----------------------------------------*/
Private int area
(
void
)
   {
   int     i;
   ULong          mask[8];
   char           r[8];


   mdlFile_parseName(geoGlobals.REF,NULL,NULL,&r[0],NULL);
   setParamsToKnownValues();


   /* Attach the reference diagram to the background */
   if (r != '\0')
      mdlRefFile_attachCoincident(r,r,NULL,NULL,TRUE,TRUE);
   mdlView_turnOn(0);


   /* Turn off views from 2 to 8 */
   for ( i = 1; i <= 7; i++)
       {
        mask[i] = 0;
        mdlView_turnOff(i);
       }
   /* Fit the diagram into the view 1 */
   mask[0] = 3;
   mdlView_fit(0,mask);
   mdlView_updateSingle(0);


   reader();
   return 0;
   }


/* ----- Set current attributes -------------------------------------- */
Public void setParamsToKnownValues(void)
   {
   double         dTemp;


   mdlCurrTrans_begin();
   mdlCurrTrans_masterUnitsIdentity (TRUE);


   mdlParams_setActive (0 , ACTIVEPARAM_LINESTYLE);
   mdlParams_setActive (0 , ACTIVEPARAM_LINEWEIGHT);
   mdlParams_setActive (17, ACTIVEPARAM_LEVEL);


   dTemp = 0.0004;
   mdlParams_setActive (&dTemp, ACTIVEPARAM_TEXTHEIGHT);
   dTemp = 0.0004;
   mdlParams_setActive (&dTemp, ACTIVEPARAM_TEXTWIDTH);
   mdlParams_setActive (TXTJUST_CB, ACTIVEPARAM_TEXTJUST);
   mdlParams_setActive (128, ACTIVEPARAM_LINELENGTH);
   dTemp = 0.00005;
   mdlParams_setActive (&dTemp, ACTIVEPARAM_LINESPACING);
   dTemp = 200;
   mdlParams_setActive (&dTemp, ACTIVEPARAM_GRIDUNITS);
   mdlParams_setActive (5, ACTIVEPARAM_GRIDREF);


   mdlCurrTrans_end ();
   }


/* ----- Read data from the database table and process the diagram --- */
Private void reader()
   {
   MSElementUnion  ellipse,e,shape,textNode;
   MSElementDescr  *patternEdPP,*edP,*txtEDP;
   Dpoint3d           center,pts[MAX_PTS],pt[MAX_POL+1];
   double         xs,ys,ma;
   int            j,k,in[MAX_POL],n,ra;
   unsigned long   i,link,nr;
   char           message[128],x[20],y[20],r[20],st[20],dbName[16];
   LinkProps         props;
   short    linkage[8];
   int                    s1, s2, s3, s4, status, dasType, linkLength;


   n = geoGlobals.pie;
   sprintf(dbName,"%s","/0");
   mdlFile_parseName(geoGlobals.DBF,NULL,NULL,&dbName[0],NULL);


   props.information = FALSE;
   props.remote      =     FALSE;
   props.modified    = FALSE;
   props.user         = TRUE;
   dasType = 1;
   xs = 0;  ys = 0;  nr = 0;


   if (mdlDB_largestMslink(&link,dbName) == SUCCESS)
   {{
   if (link > MAX_PTS) link = MAX_PTS;
   sprintf (message, "link: %d", link);
   mdlOutput_message (message);


   for (i = 0; i <= link-1; i++)
      {{{
      s1 = mdlDB_readColumn (st,dbName,i+1,DATA_SET);
      s2 = mdlDB_readColumn (x,dbName,i+1,DATA_X);
      s3 = mdlDB_readColumn (y,dbName,i+1,DATA_Y);
      s4 = mdlDB_readColumn (r,dbName,i+1,DATA_R);


         if ((s1 != SUCCESS) || (s2 != SUCCESS) ||
            (s3 != SUCCESS) || (s4 != SUCCESS))
            {
            mdlDialog_openAlert ("Bad access to the Database Table");
            break;
            }
```

```
if (atoi(st) == geoGlobals.data_set)
  {
  center.x  = atof(x)*10.0 - 1000.0;
  center.y  = 8000.0 - atof(y)*10.0;
  center.z  = 0.0;
  ra        = 2*atoi(r);
  pts[nr].x = center.x;
  pts[nr].y = center.y;
  pts[nr].z = 0.0;
  xs        = xs + pts[nr].x;
  ys        = ys + pts[nr].y;
  if (mdlEllipse_create
             (&ellipse, NULL, &center, ra, ra, NULL, -1) == SUCCESS)
        {
        mdlElement_display (&ellipse, NORMALDRAW);
    if ((status = mdlDB_buildLink (linkage, &linkLength,
                                  DBASE_LINKAGE, &props, dbName,
                                  i+1, dasType)) == SUCCESS)
          mdlElement_appendAttributes
                                  (&ellipse, linkLength, linkage);
    mdlElement_add (&ellipse);
    }
  nr = nr + 1;
  }
}}}


   if (nr != 0)
   {{{
xs = xs / nr;  ys = ys / nr;


for (i = 0; i <= nr-1; i++)
   {
   pts[i].x = pts[i].x - xs;
   pts[i].y = pts[i].y - ys;
   myatan2(&pts[i]);
   }


for (i = 0; i <= n-1; i++)
   {
   in[i] = -1;
   ma    =  0;
   for (j = 0; j <= nr-1; j++)
      {
      if ((pts[j].z >= 2*pi/n*i) && (pts[j].z < 2*pi/n*(i+1)))
      if ((pts[j].x * pts[j].x + pts[j].y * pts[j].y) > ma)
         {
         ma    = (pts[j].x * pts[j].x + pts[j].y * pts[j].y);
         in[i] = j;
         }
      }
   };
j = 0;
for (i = 0; i <= n-1; i++)
   {
   if (in[i] >= 0)
```

```
        {
        pt[j].x = pts[in[i]].x + xs;
        pt[j].y = pts[in[i]].y + ys;
        pt[j].z = 0;
        j++;
        }
      }


    for (i = j; i <= n; i++)
      {
      pt[i].x = pt[0].x;  pt[i].y = pt[0].y;  pt[i].z = pt[0].z;
      }


    if (mdlShape_create (&shape, NULL, pt, j+1, 1) == SUCCESS)
         {{
         mdlElement_display (&shape, NORMALDRAW);
         mdlElement_add (&shape);
      if (mdlElmdscr_new (&edP,NULL,&shape) == SUCCESS)
        {
        mdlPattern_hatch
                  (&patternEdPP,edP,NULL,NULL, (pi/4)*3,60,0,FALSE,NULL);
          mdlElmdscr_display (patternEdPP, 0, NORMALDRAW);
          if (geoGlobals.hatch) mdlElmdscr_add (patternEdPP);
          mdlElmdscr_freeAll (&patternEdPP);
        mdlElmdscr_freeAll (&edP);
        }
      }}
    mdlUtil_beep(1);
       }}}
       else
         mdlDialog_openAlert ("      This data SET is empty !");
  }}
  }


void myatan2(Dpoint3d *p)
  {
  double          x,y,fi;


  x = p->x;  y = p->y;
  if (x == 0)
    {
    if (y >= 0) fi = pi/2;
    if (y < 0) fi = 3/2*pi;
    }
  if (x > 0) fi = atan(y/x);
  if (x < 0) fi = pi + atan(y/x);
  if (fi < 0) fi = 2*pi + fi;
  p->z = fi;
  }
```