MAPINFO GRID ENGINE

MapBasic scripts

using

MIGRID.DLL functions

Jacques Paris

September 2001

This document contains 4 MapBasic code listings showing how to use calls to the MiGrid library. These examples have been in the largest part extracted from the GridTools.mb listing offered by Andrew Dressel (MI Corp). I have trimmed the code and made each part an autonomous program, which explains some redundancies. I have also added comments where I judged it necessary if it was not yet included in the original listing.

The main reason for “exploding” GridTools into several parts is essentially pedagogical; the examples show more clearly than in a long listing the way to handle the functions; they also have increasing degrees of complexity and they build in part on each other.

The first two examples demonstrate that a file does not have to be open as a MI table for the handle and the file info to be retrieved (a difference with MI that “recognizes” a table only if it is open)

Ex01: the simplest example shows how to get the handle of a grid file.

Ex02: once a grid file is open, the “geographical” information is retrieved. That info goes beyond the simple image coordsys and the extents of the “map”; it covers also the number of cells and the max and min Z values. The actual size of a cell can be calculated from those variables.

Ex03: with the “geo” information of the file known, it is simple to retrieve the actual Z value of any cell. Ex03 gives the Z value at the position where the user clicked.

Ex04: a grid is first created from some info collected from the user (number of cells in X and Y, mini and maxi of z values) and some fixed (but a priori) data (projection, geographical range). A formula (also a priori) transforms the z range in equal increments based on the number of cells and assigns a value to each cell; color is applied ranging from (the last a priori) blue to red.

For comments: jacques@paris-pc-gis.com

Ex01

Demo program showing how to get the handle of a MIG grid file

```
Declare Function GE_OpenGrid lib "migrid.dll"
    (lpzFilename As string, byval cache as integer, hgrid as integer
    ) As logical

dim filein as string
dim hGrid as integer
dim lReturn as logical

filein=fileopendlg("","","MIG","Select MI Grid file")
if filein="" then exit sub end if

  lReturn = GE_OpenGrid(filein, 1024, hGrid)
  If Not lReturn Then
    Note "Open " + filein + " failed"
    Exit Sub
  End If
  If hGrid = 0 Then
    Note "Open " + filein + " failed: grid handle = 0"
    Exit Sub
  End If
Print "  Opened " + filein + chr$(10) + " with handle " + hGrid
```

Ex02

Demo program showing how to retrieve all the available "geographic information" contained in a MIG file

```
Declare Function GE_OpenGrid lib "migrid.dll"
    (lpzFilename As string, byval cache as integer, hgrid as integer
    ) As logical

Declare Function GE_GetCoordSysInfo Lib "Migrid.dll"
    (ByVal hGrid As Integer, aCoordSys As String, MinXVal As Float,
    MinYVal As Float, MaxXVal As Float, MaxYVal As Float
    ) As Logical

Declare Function GE_GetContinuousMinMax Lib "Migrid.dll"
    (ByVal hGrid As Integer, MinZVal As Float,MaxZVal As Float
    ) As Logical

Declare Function GE_GetDimensions Lib "Migrid.dll"
    (ByVal hGrid As Integer, Width As Integer, Height As Integer
    ) As Logical


dim filein, aCoordSys as string
dim hGrid as integer
dim lReturn as logical
dim MinXVal, MaxXVal, MinYVal, MaxYVal, MinZVal, MaxZVal as float
dim Width, Height as integer

filein=fileopendlg("","","MIG","Select MI Grid file")
if filein="" then exit sub end if

' open grid with GE_OpenGrid

  lReturn = GE_OpenGrid(filein, 1024, hGrid)
  If Not lReturn Then
    Note "Open " + filein + " failed"
    Exit Sub
  End If
  If hGrid = 0 Then
    Note "Open " + filein + " failed: grid handle = 0"
    Exit Sub
  End If
print "FILE  "+filein

' geographical parameters with GE_GetCoordSysInfo
' initialize the string variable to allocate actually memory

  aCoordSys = Space$(255)
  lReturn = GE_GetCoordSysInfo(hGrid, aCoordSys, MinXVal, MinYVal,
        MaxXVal, MaxYVal)
    Print "  " + aCoordSys
    Print "  MinXVal = " + MinXVal + ", MinYVal = " + MinYVal +
       chr$(10)+ "  MaxXVal = " + MaxXVal + ", MaxYVal = " + MaxYVal
```

```
' Get minimum and maximum grid values with GE_GetContinuousMinMax

  lReturn = GE_GetContinuousMinMax(hGrid, MinZVal, MaxZVal)
    Print "  MinZVal = " + MinZVal + ", MaxZVal = " + MaxZVal

' Get grid dimensions (n of rows and columns) with GE_GetDimensions

  lReturn = GE_GetDimensions(hGrid, Width, Height)
    Print "  Width (in cells) = " + Width + ", Height = " + Height
```

Ex03

Demo program showing how to retrieve data from a MIG grid file

```
include "mapbasic.def"
include "icons.def"

Declare Function GE_OpenGrid lib "migrid.dll"
   (filein As string, byval cache as integer, hgrid as integer
   ) As logical

Declare Function GE_GetCoordSysInfo lib "migrid.dll"
   (ByVal hgrid as integer, clau as string, mix as float, miy as float,
   max as float, may as float
   ) as logical

Declare Function GE_GetContinuousMinMax lib "migrid.dll"
   (ByVal hgrid as integer, zmin as float, zmax as float
   ) as logical

Declare Function GE_GetDimensions lib "migrid.dll"
   (ByVal hgrid as integer, wid as integer, hei as integer
   ) as logical

Declare Function GE_GetContinuousValue lib "migrid.dll"
   (ByVal hgrid as integer,ByVal icol as integer,ByVal jrow as integer,
   zval as float, ax as smallint
   ) as logical

Declare Function GE_GetGridType lib "migrid.dll"
   (ByVal hgrid as integer, gtyp as smallint) as logical

Declare Function GE_StartRead lib "migrid.dll"
   (ByVal hgrid as integer) as logical

Declare Function GE_EndRead lib "migrid.dll"
   (ByVal hgrid as integer) as logical

Declare Function GE_CloseGrid lib "migrid.dll"
   (hgrid as integer) as logical

declare sub main
declare sub GridInfoToolHandler
declare function findlayer as string
declare sub about
declare sub goodbye

dim filein, claus as string
dim typg, puchIsNull as smallint
dim hgrid, widt, heig ,icol,jrow as integer
dim xmin,ymin,xmax,ymax,zmin,zmax, zval as float
dim iret as logical
```

```
'======
sub main
'======

' In this program, a menu is useful mainly to terminate the application
' and remove the tool from the ToolPad. Also an opportunity to say
' something about the program

create menu "Read Grid" as
    "About Read Grid" calling about,
    "Remove grid Grid" calling goodbye

alter menu id 4 add "Read Grid" as "Read Grid"

   Alter ButtonPad ID 3
    Add
      Separator
      ToolButton ID 9944
        Calling GridInfoToolHandler
        Icon 46
        Cursor MI_CURSOR_CROSSHAIR
        DrawMode  DM_CUSTOM_POINT
        HelpMsg "Retrieve value from grid cell.\nRetrieve grid value"
    Show
end sub

'======
Sub GridInfoToolHandler
'======

  OnError Goto HandleError

  Dim sCmd, val As String
  Dim MapWindowID, lCol, lRow  As Integer
  Dim x, y, xx, yy, pdvalue As Float
  Dim i, puchIsNull As SmallInt

' check where click was located and identify the table (=layer)

  If WindowInfo( frontwindow(), WIN_INFO_TYPE) <> WIN_MAPPER Then
    Note "Click in a mapper with a MIG layer."
    Exit Sub
  End If
filein=findlayer()

' retrieve geographic info for the grid

claus=space$(255)
iret=GE_GetCoordSysInfo(hgrid,claus,xmin,ymin,xmax,ymax)
iret=GE_GetDimensions(hgrid,widt,heig)
iret=GE_GetGridType(hgrid,typg)
iret=GE_GetContinuousMinMax(hgrid,zmin,zmax)

' retrieve cursor position and open grid file

  sCmd = "Set " + claus
  Run Command sCmd
  x = CommandInfo(CMD_INFO_X)
  y = CommandInfo(CMD_INFO_Y)
  iret = GE_StartRead(hGrid)
if not iret then
note filein+"  with handle "+hgrid+chr$(10)+chr$(10)+"cannot be open
for reading"
exit sub
end if

' convert cursor position in geo coordinates
' and extract corresponding z value

  If iret Then
    lCol = (Widt * (x - XMin) / (XMax - XMin)) + .5
    lRow = (Heig - Heig * (y - YMin) / (yMax - YMin)) + .5
   xx=XMin+(XMax-XMin)/Widt*(lcol-.5)
   yy=Ymax-(YMax-YMin)/Heig*(lrow-.5)
   iret = GE_GetContinuousValue(hGrid, lCol, lRow, pdValue, puchIsNull)

    If lCol < 0 Or lRow < 0 Or lCol >= Widt Or lRow >= Heig Then

' the user clicked in a map but outside the bounds of a mig file

      If pdValue = 0 Then
       val=" is undefined"
      Else
       val=str$(pdvalue)+ "  but should be undefined"
      End If
    Else
      If puchIsNull Then

' distinguish between zero values and NULL

       val="NULL."
      Else
       val=str$(pdValue)
      End If
    End If

dialog title "Information on a cell in a MIG file"
   control statictext title "File : " position 10,10
   control statictext title filein position 10,20
   control statictext title "Detected cell position" position 10,35
   control statictext title "Row : "+lrow position 20,45
   control statictext title "Column : "+lcol position 80,45
   control statictext title "Estimated cell centroid coordinates"
position 10,60
   control statictext title "X : "+xx position 20,70
   control statictext title "Y : "+yy position 80,70
   control statictext title "Value : "+val position 45,90
   control okbutton position 50,110

' terminate read of grid

    iret = GE_EndRead(hGrid)
  Else
    Note "  StartRead(" + hGrid + ") failed"
  End If

' close grid

  iret = GE_CloseGrid(hGrid)

  Exit Sub

HandleError:
  Note "GridInfoToolHandler: " + Error$()
  Resume Next

end sub

'======
function findlayer as string
'======

dim spath as string
dim i as smallint
dim MapWindowID as integer

MapWindowID=frontwindow()

' find the first layer in the pile that is a GRID file

  For i = 1 To MapperInfo(MapWindowID, MAPPER_INFO_LAYERS)
    If LayerInfo(MapWindowID, i, LAYER_INFO_TYPE) =
       LAYER_INFO_TYPE_GRID Then
      sPath = LayerInfo(MapWindowID, i, LAYER_INFO_PATH)
      Exit For
    End If
  Next

' find if the corresponding file exists

 sPath = Left$(sPath, Len(sPath)-3) + "mig"
  If Not FileExists(sPath) Then
    Note "Cannot find grid file " + sPath
    Exit function
  End If

' get the file handle

  iret = GE_OpenGrid(sPath, 1024, hGrid)

' prepare function result

if not iret then
note "Open failed on file   "+spath
goto faux
end if
if hgrid =0 then
note "Handle of ZERO on file   "+spath
goto faux
end if
```

```
findlayer=spath
exit function

faux:
findlayer=""

end function

'======
sub about
'======

note "Read Grid reads the z value of the cell in whcih the user
clicked. "+
       "Click on 'arrow+?' icon added in the tool bar then the
CrossHair cursor"+chr$(10)+chr$(10)+
       "Demo program part of 'GridEngine calls in MapBasic', J.Paris,
Sept 01"
end sub

'======
sub goodbye
'======

end program
end sub
```

Ex04

Demo program showing how to create a MIG grid file and write data to it

```
include "mapbasic.def"

Define ID_EDIT_TEXT_FILENAME 601
Define ID_EDIT_TEXT_ROWS 602
Define ID_EDIT_TEXT_COLS 603
Define ID_EDIT_TEXT_MIN 604
Define ID_EDIT_TEXT_MAX 605

Define GE_GRIDINFO_MAGIC_NUMBER 13124
Define GE_GRIDINFO_INVALID      43690
Define GE_GRIDTYPE_CONTINUOUS   1
Define GE_GRIDTYPE_CLASSIFIED   2
Define GE_MAX_INFLECTIONS 255
Define _MAX_PATH 260                      ' max. length of full
pathname
Define GE_COLOR Integer
Define GE_HGRID Integer

Type GE_COLORINFLECTIONS
  sNumInflections As SmallInt
  alignmentfiller(3) As SmallInt
  adValue(GE_MAX_INFLECTIONS) As FLoat
  aColor(GE_MAX_INFLECTIONS) As GE_COLOR
End Type

Type GE_GRID_INFO
  lMagic As Integer
  lWidth As Integer
  lLength As Integer
  ptchCoordSys As String
  dMinXVal As Float
  dMaxXVal As Float
  dMinYVal As Float
  dMaxYVal As Float
End Type

Declare Function GE_GetDefaultWriteHandler Lib "Migrid.dll"
    (ByVal sGridType As SmallInt, ptchHandlerName As String
    ) As Logical

Declare Function GE_CreateContinuousGrid Lib "Migrid.dll"
    (ptchHandlerName As String,  ptchFilename As String,
    pInflections As GE_COLORINFLECTIONS,
    ByVal uchIsNullTransparent As SmallInt, clrNull As GE_COLOR,
    pGridInfo  As GE_GRID_INFO, ByVal dMinVal As Float,
    ByVal dMaxVal As FLoat, phGrid As GE_HGRID
    ) As Logical

Declare Function GE_WriteContinuousValue Lib "Migrid.dll"
    (ByVal hGrid As GE_HGRID, ByVal lCol As Integer,
    ByVal lRow As Integer, ByVal dValue As Float
    ) As Logical
```

```
Declare Function GE_CloseContinuousGrid Lib "Migrid.dll"
   (phGrid As GE_HGRID) As Logical


declare sub main
Declare Function CreateGridDialog() As Logical
Declare Sub BrowseButtonHandler
Declare Sub OKButtonHandler

Global szGridFilename As String
Global iRows, iCols As Integer
Global fMin, fMax As Float

'======
sub main
'======

Dim ret As Logical
Dim atchHandlerName As String
Dim hGrid As GE_HGRID
Dim Inflections As GE_COLORINFLECTIONS
Dim GridInfo As GE_GRID_INFO
Dim clrNull As GE_COLOR
Dim uchIsNullTransparent As SmallInt
Dim r, c, i As Integer
Dim dValue As Float

' get the default grid handler name.
' initialize first the string variable holding the name

atchHandlerName = Space$(_MAX_PATH)
ret = GE_GetDefaultWriteHandler(GE_GRIDTYPE_CONTINUOUS,
atchHandlerName)
print chr$(12)
print "GetDefaultWriteHandler returned with " + atchHandlerName

' call the dialog to input grid parameters

 If Not CreateGridDialog() Then Exit Sub  End If

' set the various parameters for the "coloring"
' there will be continuous coloring from BLUE to RED
' the "end points" are counted as inflections.
' to add another intermediate point at green we could have
' inflections.snuminflections =3
' the first point with no change, the previous (2) renamed (3) and
'   Inflections.adValue(2) = (fMax - fMin)/2
'   Inflections.aColor(2) = RGB(0,255,0)

  Inflections.sNumInflections = 2
  Inflections.adValue(1) = fMin
  Inflections.aColor(1) = RGB(0,0,255)
  Inflections.adValue(2) = fMax
  Inflections.aColor(2) = RGB(255,0,0)

' a simple lon/lat "map will be produced with a one degree span in both
' directions

  GridInfo.lMagic = GE_GRIDINFO_MAGIC_NUMBER
  GridInfo.lWidth = iCols
  GridInfo.lLength = iRows
  GridInfo.ptchCoordSys = "CoordSys Earth Projection 1, 62"
  GridInfo.dMinXVal = 1
  GridInfo.dMaxXVal = 2
  GridInfo.dMinYVal = 1
  GridInfo.dMaxYVal = 2

  clrNull = RGB(0,0,0)
  uchIsNullTransparent = 1     ' 0=opaque, 1=transparent

' if more than 2 inflections, Inflections.adValue(2) >...Value(number
' of inflections) the "frame" of the grid is created but the z values
' are not entered

  ret = GE_CreateContinuousGrid(atchHandlerName, szGridFilename,
Inflections, uchIsNullTransparent,
     clrNull, GridInfo, Inflections.adValue(1), Inflections.adValue(2),
hGrid)

  print "  Created Continuous Grid " + szGridFilename

' the z values are calculated as proportions of the defined Z range
' given the order of creation of the value from the upper left to the
' lower right corner. This formula is just a demo of value generation.

  For r=0 To GridInfo.lLength-1
    For c=0 To GridInfo.lWidth-1
      dValue = Inflections.adValue(1) + (Inflections.adValue(2)-
Inflections.adValue(1)) *
                  ((r*GridInfo.lWidth+c) / (GridInfo.lLength *
GridInfo.lWidth))
        Print "  Row:"+r+" Col:"+c+" Val="+dValue
      ret = GE_WriteContinuousValue(hGrid, c, r, dValue)
    Next
  Next

' closing the grid

  ret = GE_CloseContinuousGrid(hGrid)
  print "Continuous Grid Closed"

'registering the grid as a MI table

    Register Table TrueFileName$(szGridFileName) Type "GRID"
    Open Table Left$(szGridFileName, Len(szGridFileName)-4)
    Map From TableInfo(0, TAB_INFO_NAME)

end sub

'=======
Function CreateGridDialog() As Logical
'=======
Dim sRows, sCols, sMin, sMax As String

' read in the four variables required from the user

sRows = Str$(iRows)
sCols = Str$(iCols)
sMin = Str$(fMin)
sMax = Str$(fMax)

Dialog Title "Create a new grid file"
    Control StaticText Title "Grid File Name:"  Position 10, 12
    Control EditText Value szGridFilename Into szGridFilename ID
ID_EDIT_TEXT_FILENAME
      Position 60, 10 Width 200
    Control Button  Title "&Browse..." Calling BrowseButtonHandler
Position 270, 10
    Control StaticText Title "Rows:" Position 10, 32
    Control EditText Value sRows Into sRows ID ID_EDIT_TEXT_ROWS
Position 40, 30
    Control StaticText Title "Columns:" Position 10, 47
    Control EditText Value sCols Into sCols ID ID_EDIT_TEXT_COLS
Position 40, 45
    Control StaticText Title "Minimum value:" Position 143, 32
    Control EditText Value sMin Into sMin ID ID_EDIT_TEXT_MIN Position
195, 30
    Control StaticText Title "Maximum value:" Position 143, 47
    Control EditText Value sMax Into sMax ID ID_EDIT_TEXT_MAX Position
195, 45
    Control OKButton Title "&OK" Position 100, 90 Calling
OKButtonHandler
    Control CancelButton Title "&Cancel" Position 150, 90

 If CommandInfo(CMD_INFO_DLG_OK) Then
    iRows = Val(sRows)
    iCols = Val(sCols)
    fMin = Val(sMin)
    fMax = Val(sMax)
    CreateGridDialog = TRUE
  Else
    CreateGridDialog = FALSE
  End If

End Function

'======
Sub BrowseButtonHandler
'======
' retrieve the path of the selected file

 szGridFilename = FileSaveAsDlg(PathToDirectory$(TempFileName$("")),"",
       "MIG", Specify grid file name")
  If szGridFileName <> "" Then
    Alter Control ID_EDIT_TEXT_FILENAME
      Value szGridFileName
  End If

End Sub
```

```
'======
Sub OKButtonHandler
'======
' an added security. Verify that the grid file has been properly
' created

Dim sRows, sCols As String
Dim i As Integer

szGridFilename = ReadControlValue(ID_EDIT_TEXT_FILENAME)

If szGridFilename <> "" Then
   If Right$(UCase$(szGridFilename),4) <> ".MIG" Then
      i = InStr(1, szGridFilename, ".")
      If i > 0 Then
       szGridFilename = Left$(szGridFilename, i-1)
      End If
      szGridFilename = szGridFilename + ".MIG"
      Open File szGridFilename For Output As #1
      Close File #1
      Alter Control ID_EDIT_TEXT_FILENAME Value szGridFilename
   End If
  Else
   Note "Invalid blank grid file name"
   Dialog Preserve
End If

sRows = ReadControlValue(ID_EDIT_TEXT_ROWS)
sCols = ReadControlValue(ID_EDIT_TEXT_COLS)
iRows = Val(sRows)
iCols = Val(sCols)
If iRows < 1 Or iCols < 1 Then
   Note "Rows and Columns need to be greater than 0"
   Dialog Preserve
End If

End Sub
```