รายการอ้างอิง

ภาษาไทย

1. ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ Visual Basic for Windows 3.0 บริษัท สกายบุ๊กส์จำกัด 2538

2 ณรงค์ ชอนตะวัน "การศึกษาและการออกแบบสร้างเครื่องนับรอยรังสีอัลฟาบนฟิลม์" วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย , 2523.

ภาษาอังกฤษ

3. Asus 3DexPlorer GX2 User's manual : Asustek Computer inc. 1997

4 Craig A.Lindley Practical Image Processing in C : John Wiley & Sons, Inc. 1991

5. Eichholz, G.G. ; and Poston, J.W. Principle of Nuclear Radiations Detection. Michigan: Ann Arbar Science Publishers, Inc., 1979.

6. Fleischer, R. L., P.B. and Walker, R.M. Nuclear Track in Solids. Berkey : University of California Press, 1975

7. Francois, H., Kurtz,N.,Massue,J.P.,Monnin, M.,Schmitt,R. and Durrani, S. A.(eds.) Solid State Nuclear Track Detectors. Oxford : Pergamon Press,1980.

8. Gardner, R.P. and Ely, R.L. Jr. Radioisotope Measurement Applications in Engineering. NewYork : Reinhole Publishing Corporation, 1967.

9. Knoll, G.F. Radiation Detection and Mesurement. New York: John Wiley and Sons, 1977

10. Knop, G. and Paul, W, Alpha, Beta, and Gamma-rays Spectroscopy Volume I. Edited by Kai Seigbhan. Amstersam : North Holland Publishing Co., 1975.

11. Parker, James R. Algorithms for Image Processing and Computer Vision : Wiley Computer Publishng 1997 .

12. Peter G. Aitken Visual Basic for Windows 95 Insidser : John Wiley & Sons, Inc. , 1996.

13. Pilcher, V.E. Jones, C.C. and Ellmer, G.R. "Particle Tracks in Cellulose Nitrate. " American Journal of Physics 40 (May 1972) : 679-683.

14. Qaqish, A.Y.; and Besand, C.B. "Detection Efficiency and Range Determination of Alpha Particles in Cellulose Nitrate." Nucl. Intru. and Meth. 138(1976): 493-505

15. Rod Stephens Visual Basic Graphics Programming : Wiley Computer Publishing, 1997.

ภาคผนวก

ภาคผนวก ก.

การใช้งานโปรแกรมนับภาพรอยอนุภาคแอลฟา

การติดตั้งโปรแกรม โปรแกรมที่ถูกสร้างขึ้น สามารถใช้งานได้บนระบบปฏิบัติการ Microsoft Windows 3.1 และ Windows 95 ตัวโปรแกรมจะประกอบไปด้วยไฟล์ต่างๆดังนี้

1. โปรแกรมหลัก ไฟล์ชื่อ Project.exe
2. ไฟล์ Dynamic Link Libraly dll1.dll
3. ไฟล์ Dynamic Link Libraly vb40016.dll
4. ไฟล์ Dynamic Link Libraly bc450rtl.dll
5. ไฟล์ Dynamic Link Libraly ddl1wpj.dll
6. ไฟล์ Dynamic Link Libraly oc25.dll
7. ไฟล์ Dynamic Link Libraly dao2516.dll
8. ไฟล์ comdlg16.ocx
8. ไฟล์ bright.dib

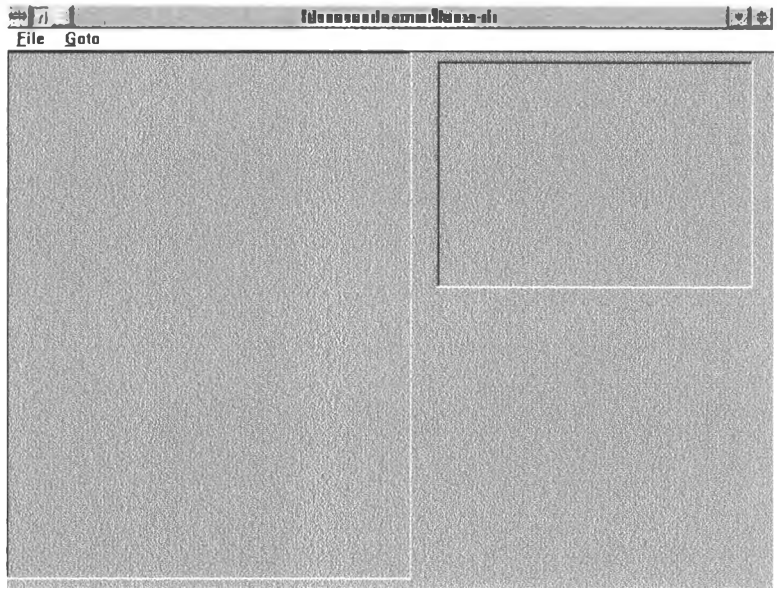
ระบบที่โปรแกรมต้องการมีดังนี้

1. ไมโครคอมพิวเตอร์ ที่มี ตัวประมวลผลกลาง 80486 ขึ้นไป
2. เนื้อที่ Hard disk ประมาณ 6 megabytes
3. ระบบแสดงภาพ ที่ 640x480 256 สี หรือ 800x600 256 สี
4. ระบบปฏิบัติ Windows 3.11 หรือ Windows 95
5. หน่วยความจำ 8 megabytes ขึ้นไป

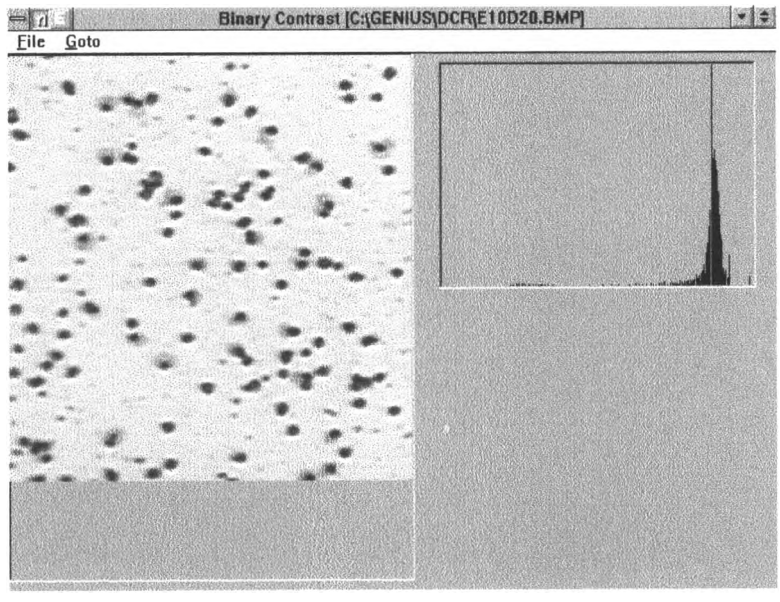
การติดตั้งโปรแกรมมีขั้นตอนดังนี้

1. สร้าง directory c:\project โดยใช้คำสั่งใน dos คือ md project
2. ทำการคัดลอก ไฟล์ project.exe, dll1.dll, bc450rtl.dll , ddl1wpj.dll และbright.dib ไปไว้ใน directory c:\project
3. ทำการคัดลอกไฟล์ vb40016.dll , oc25.dll, dao2516.dll และ comdlg16.ocx ไปไว้ใน directory c:\windows\system
4. เข้าโปรแกรม windows แล้วใช้คำสั่ง file run c:\sketchup\project.exe โปรแกรมก็จะทำงานดังรูปที่ ก1

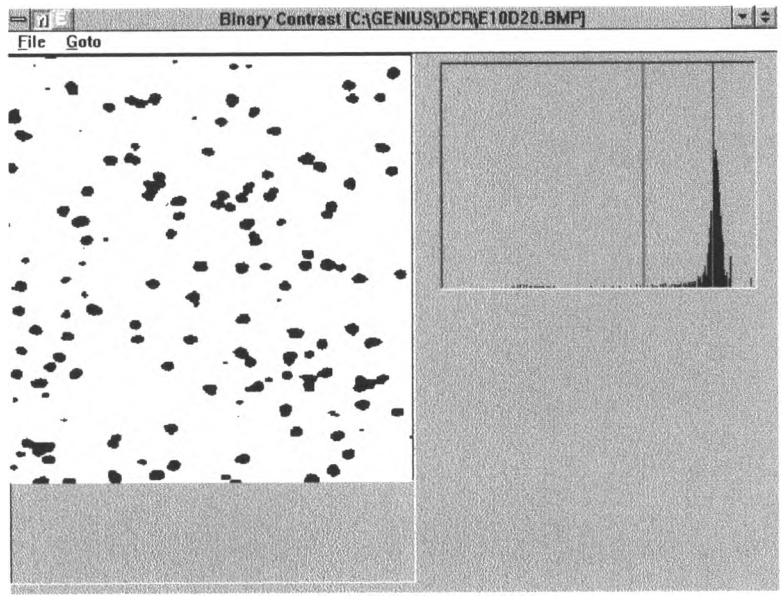
รูปที่ ก1 แสดงหน้าจอเมื่อโปรแกรมนับรอยอนุภาคแอลฟาถูกเรียกเข้ามาใช้งาน



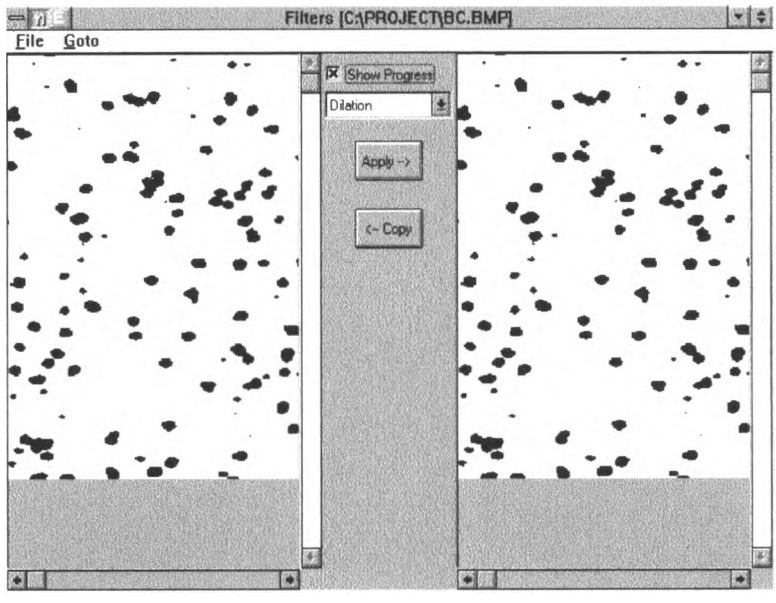
รูปที่ ก2 แสดงหน้าจอเมื่อโปรแกรม Binary Contrast ถูกเรียกใช้งาน



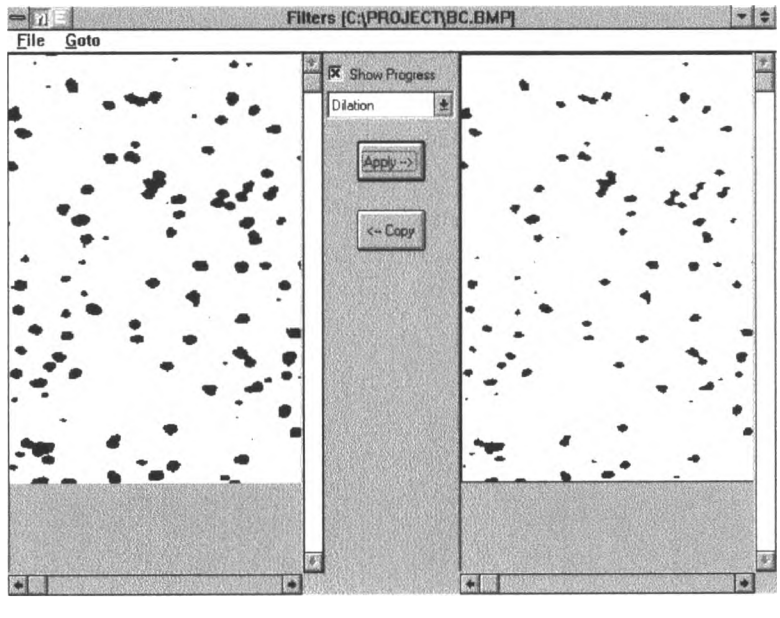
รูปที่ ก3 แสดงหน้าจอเมื่อเรียกภาพรอยอนุภาคที่ได้จากล้องจุลทรรศน์



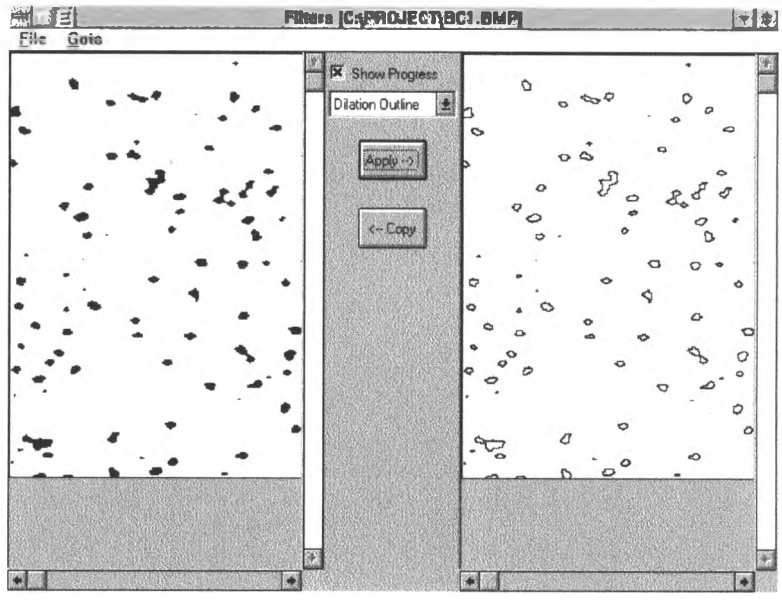
รูปที่ ก4 แสดงหน้าจอเมื่อแปลงภาพรอยอนุภาคเป็นภาพขาว-ดำ



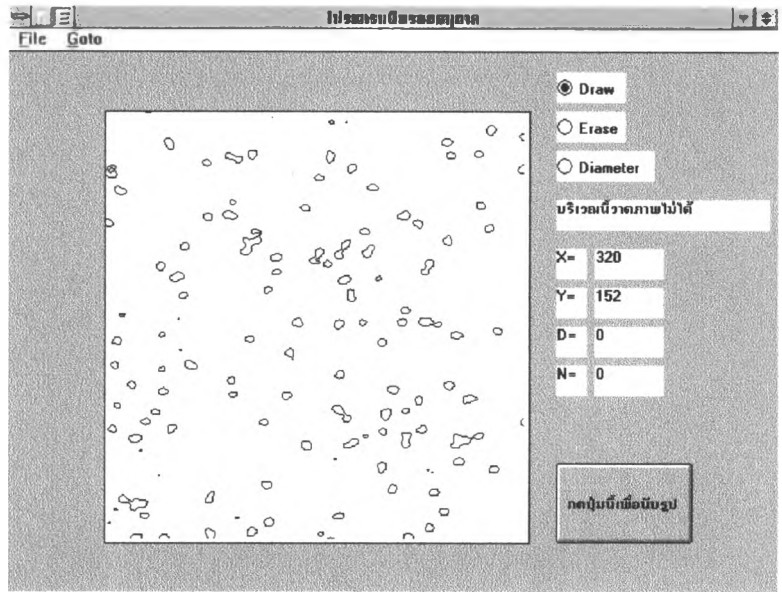
รูปที่ ก5 แสดงโปรแกรมตกแต่งภาพเข้าใช้งาน



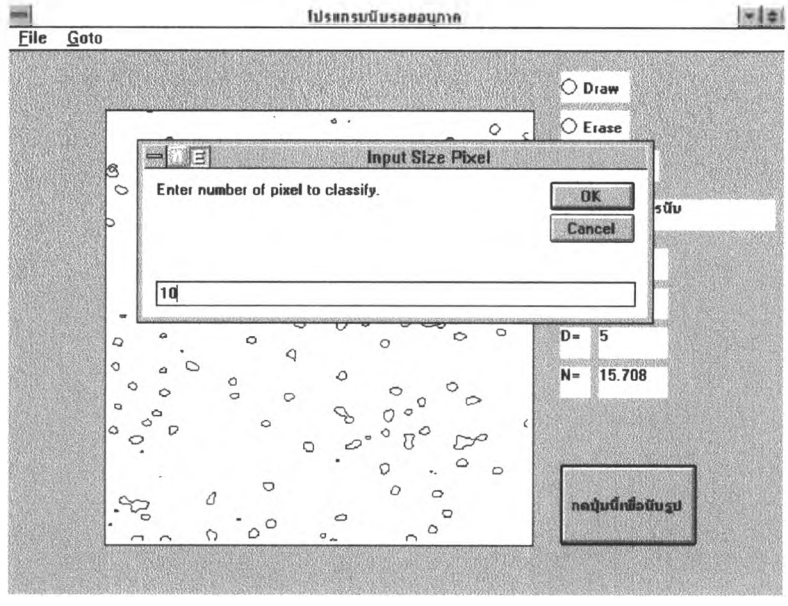
รูปที่ ก6 แสดงภาพรอยอนุภาคที่ผ่านการทำ dilation เพื่อกำจัดจุดภาพที่ไม่ต้องการออกไป



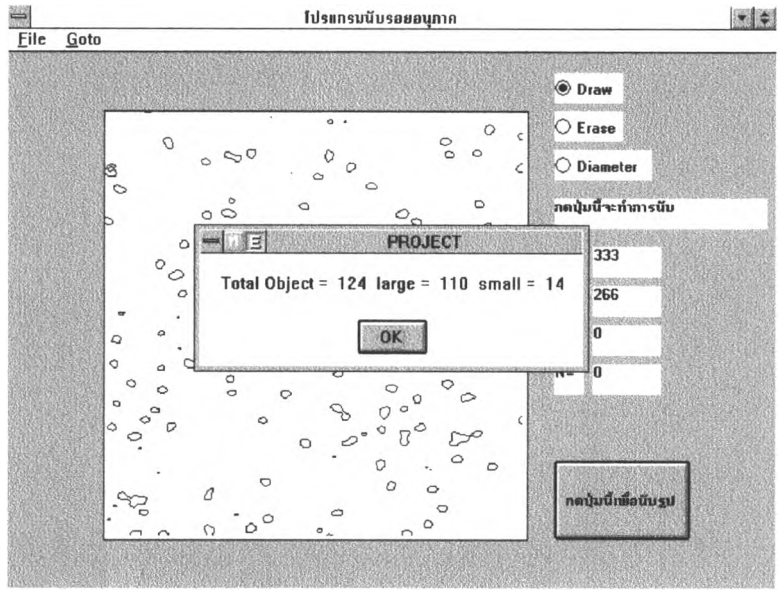
รูปที่ ก7 แสดงภาพรอยอนุภาคที่ผ่านการทำ dilation outlining



รูปที่ ก8 แสดงหน้าจอเมื่อโปรแกรมนับรอยถูกเรียกเข้าใช้งาน



รูปที่ ก9 แสดงการเลือกขนาดของรอยที่ต้องการให้โปรแกรมนับ
ในที่นี้เลือก 10 หมายถึงให้โปรแกรมนับรอยที่มีจุดรอบรูป
ตั้งแต่ 10 จุดขึ้นไป



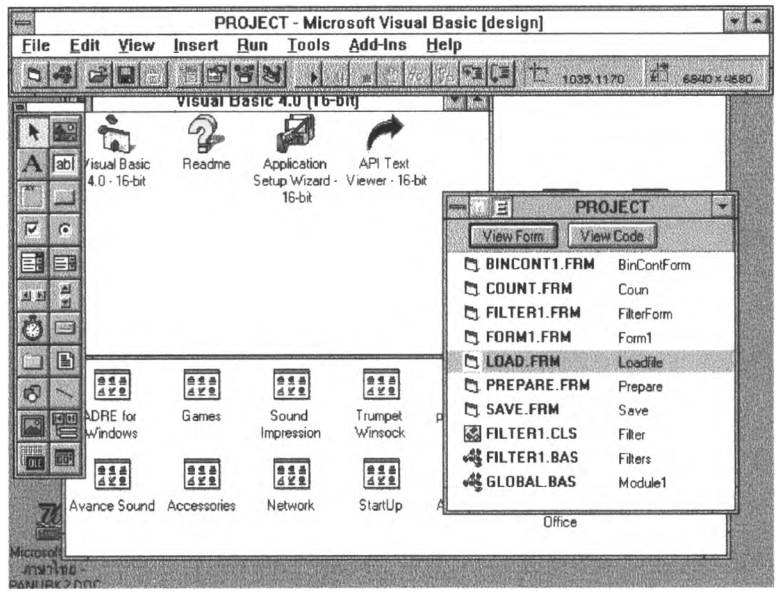
รูปที่ ก10 แสดงผลการนับรอยอนุภาค จะแสดงจำนวนรอยทั้งหมด
จำนวนรอยที่มีขนาดจุดมากกว่า 10 จุดขึ้นไป

ภาคผนวก ข.

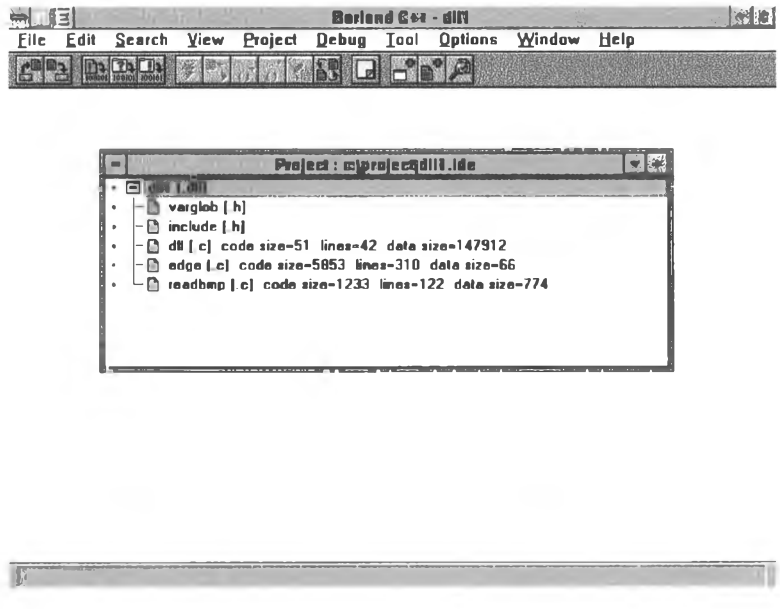
แสดงรายละเอียดของโปรแกรม

รายละเอียดของโปรแกรมนับภาพจะประกอบด้วยไฟล์ต่างๆดังนี้

1. ไฟล์ project.mak เป็นไฟล์ของหลักที่จะเรียกไฟล์อื่นๆเข้าใช้งานต่อไป
2. ไฟล์ form.frm เป็นแบบฟอร์มเริ่มต้นของโปรแกรม
3. ไฟล์ bincont1.frm เป็นฟอร์มแปลงภาพเป็นขาว-ดำ
4. ไฟล์ filter1.frm เป็นฟอร์มตกแต่งภาพ
5. ไฟล์ global.bas เป็นการกำหนดตัวแปรในโปรแกรม
6. ไฟล์ prepare.frm เป็นฟอร์มที่จะเรียกภาพที่จะทำการนับเข้าสู่โปรแกรม
7. ไฟล์ count.frm เป็นฟอร์มที่จะเรียก dll1.dll ซึ่งประกอบด้วย readbmp.c และ edge.c เข้าใช้งาน
8. ไฟล์ load.frm เป็นฟอร์ม เรียกภาพ
9. ไฟล์ save.frm เป็นฟอร์มจัดเก็บภาพ
10. ไฟล์ dll1.ide เป็นไฟล์ project ที่พัฒนาขึ้นโดยใช้ Borland C++ 4.5 for windows 3.1 การคอมไฟล์ให้เรียก project dll1.ide แล้วคอมไฟล์โดยใช้คำสั่ง make all
11. ไฟล์ readbmp.c เป็นไฟล์ที่ทำการแปลงภาพให้เป็นแบบ bit matrix จะถูกคอมไพล์และรวมอยู่ในไฟล์ dll1.dll
12. ไฟล์ edge.c เป็นไฟล์ที่ใช้ติดตามขอบภาพ พร้อม เก็บข้อมูลขอบภาพ จะถูกคอมไพล์และรวมอยู่ในไฟล์ dll1.dll



รูปที่ ข1 แสดงไฟล์ทั้งหมดใน project.vbp



รูปที่ ข2 แสดงไฟล์ทั้งหมดใน project dll1.ide

ไฟล์ project.vbp

```
Form=FORM1.FRM
Module=Filters; FILTER1.BAS
Class=Filter; FILTER1.CLS
Module=BinCont; BINCONT1.BAS
Module=Module1; GLOBAL.BAS
Form=BINCONT1.FRM
Form=FILTER1.FRM
Form=PREPARE.FRM
Form=SAVE.FRM
Form=LOAD.FRM
Form=COUNT.FRM
Reference=*\G{BEF6E001-A874-101A-8BBA-00AA00300CAB}#1.0#0#C:\WINDOWS\SYSTEM\oc25.dll#Standard OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#2.5#0#C:\WINDOWS\SYSTEM\dao2516.dll#Microsoft DAO 2.5 Object Library
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMDLG16.OCX
ProjWinSize=153,363,248,215
ProjWinShow=2
IconForm=" "
HelpFile=""
ExeName32="Project.exe"
Path32="C:\PROJECT"
ExeName="PROJECT.EXE"
Name="Project"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
VersionCompatible="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="  "
```

ไฟล์ prepare.frm

```
VERSION 4.00
Begin VB.Form Prepare
   Appearance      =  0  'Flat
   BackColor       =  &H00C0C0C0&
   Caption         =  "โปรแกรมนับรอยอนุภาค"
   ClientHeight    =  6630
   ClientLeft      =  75
   ClientTop       =  1095
   ClientWidth     =  9600
   BeginProperty Font
      name            =  "MS Sans Serif"
      charset         =  1
      weight          =  700
      size            =  8.25
      underline       =  0  'False
      italic          =  0  'False
      strikethrough   =  0  'False
   EndProperty
   ForeColor       =  &H80000008&
   Height          =  7320
   Left            =  15
   LinkMode        =  1  'Source
   LinkTopic       =  "Form1"
   ScaleHeight     =  36
   ScaleMode       =  3  'Pixel
   ScaleWidth      =  36
   Top             =  465
   Width           =  9720
   WindowState     =  2  'Maximized
   Begin VB.TextBox Text4
      Appearance      =  0  'Flat
      BorderStyle     =  0  'None
      Height          =  375
      Left            =  6840
      TabIndex        =  13
      Text            =  "N="
      Top             =  3840
      Width           =  375
   End
   Begin VB.TextBox Text3
      Appearance      =  0  'Flat
      BorderStyle     =  0  'None
      Height          =  375
      Left            =  6840
      TabIndex        =  11
      Text            =  "D="
      Top             =  3360
```

```
   Width          =  375
End
Begin VB.TextBox Text2
   Appearance     =  0  'Flat
   BorderStyle    =  0  'None
   Height         =  375
   Left           =  6840
   TabIndex       =  9
   Text           =  "Y="
   Top            =  2880
   Width          =  375
End
Begin VB.TextBox Text1
   Appearance     =  0  'Flat
   BorderStyle    =  0  'None
   Height         =  375
   Left           =  6840
   TabIndex       =  8
   Text           =  "X="
   Top            =  2400
   Width          =  375
End
Begin VB.OptionButton Option3
   Appearance     =  0  'Flat
   BackColor      =  &H80000005&
   Caption        =  "Diameter"
   ForeColor      =  &H80000008&
   Height         =  375
   Left           =  6840
   TabIndex       =  5
   Top            =  1200
   Width          =  1215
End
Begin VB.CommandButton Command1
   Appearance     =  0  'Flat
   BackColor      =  &H80000005&
   Caption        =  "กดปุ่มนี้เพื่อนับรูป"
   Height         =  975
   Left           =  6840
   TabIndex       =  3
   Top            =  5040
   Width          =  1695
End
Begin VB.OptionButton Option2
   Appearance     =  0  'Flat
   BackColor      =  &H80000005&
   Caption        =  "Erase"
   ForeColor      =  &H80000008&
   Height         =  375
```

```
   Left            =   6840
   TabIndex        =   1
   Top             =   720
   Width           =   855
End
Begin VB.OptionButton Option1
   Appearance      =   0  'Flat
   BackColor       =   &H80000005&
   Caption         =   "Draw"
   ForeColor       =   &H80000008&
   Height          =   375
   Left            =   6840
   TabIndex        =   2
   Top             =   240
   Width           =   855
End
Begin VB.PictureBox Immagine1
   Appearance      =   0  'Flat
   AutoRedraw      =   -1  'True
   BackColor       =   &H00FFFFFF&
   DrawMode        =   10  'Mask Pen
   DrawWidth       =   4
   FillStyle       =   0  'Solid
   ForeColor       =   &H80000008&
   Height          =   5310
   Left            =   1200
   MousePointer    =   2  'Cross
   Picture         =   "PREPARE.frx":0000
   ScaleHeight     =   352
   ScaleMode       =   0  'User
   ScaleWidth      =   352
   TabIndex        =   0
   Top             =   720
   Width           =   5310
End
Begin VB.Label Label5
   Appearance      =   0  'Flat
   BackColor       =   &H80000005&
   Caption         =   "Label5"
   ForeColor       =   &H80000008&
   Height          =   375
   Left            =   7320
   TabIndex        =   12
   Top             =   3840
   Width           =   855
End
Begin VB.Label Label4
   Appearance      =   0  'Flat
   BackColor       =   &H80000005&
```

```
    Caption         =   "Label4"
    ForeColor       =   &H80000008&
    Height          =   375
    Left            =   7320
    TabIndex        =   10
    Top             =   3360
    Width           =   855
  End
  Begin VB.Label Label3
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    Caption         =   "Label3"
    ForeColor       =   &H80000008&
    Height          =   375
    Left            =   7320
    TabIndex        =   7
    Top             =   2880
    Width           =   855
  End
  Begin VB.Label Label1
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    Caption         =   "Label1"
    ForeColor       =   &H80000008&
    Height          =   375
    Left            =   7320
    TabIndex        =   6
    Top             =   2400
    Width           =   855
  End
  Begin VB.Label Label2
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    Caption         =   "Label2"
    ForeColor       =   &H80000008&
    Height          =   375
    Left            =   6840
    TabIndex        =   4
    Top             =   1800
    Width           =   2655
  End
  Begin VB.Menu mnuFile
    Caption         =   "&File"
    Begin VB.Menu mnusave
      Caption       =   "&Save"
    End
    Begin VB.Menu mnuFileload
      Caption       =   "&Load"
    End
```

```
    Begin VB.Menu mnuexit
      Caption         =   "&Exit"
    End
  End
  Begin VB.Menu MnuGo
    Caption         =   "&Goto"
    Begin VB.Menu MnuBincont
      Caption         =   "&Binary Contrast Program"
    End
    Begin VB.Menu MnuFilter
      Caption         =   "&Filter Program"
    End
  End
End
Attribute VB_Name = "Prepare"
Attribute VB_Creatable = False
Attribute VB_Exposed = False

Dim Xorig, Yorig, x1, y1 As Integer
Dim Xcour, Ycour, x2, y2 As Integer
Private Declare Function preparefile Lib "C:\Project\Dll1.dll" (ByVal i As Integer) As Integer




Private Sub Command1_Click()

    SavePicture Immagine1.Image, "c:\project\px2.bmp"
    Load Coun
    Coun.Show 1

End Sub

Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

    label2.Caption = "กดปุ่มนี้จะทำการนับ"  '      label2 contains the description of the
                                                '          button "Command1"
End Sub

Private Sub Command2_Click()

    SavePicture Immagine1.Image, "c:\project\px2.bmp"
    Unload Scheda1
    prova

End Sub
```

```
Private Sub Command2_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)


End Sub

Private Sub Form_Load()

    Label4.Caption = "0"
    Label5.Caption = "0"


    If count1 = 1 Then
Immagine1.Picture = LoadPicture("c:\project\px2.bmp")

        Option1.value = True

    Else
Immagine1.Picture = LoadPicture("c:\project\bright.dib")

        Option1.value = True

    End If

End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

    label2.Caption = "บริเวณนี้วาดภาพไม่ได้"

End Sub




Private Sub Immagine1_MouseDown(pulsante As Integer, Shift As Integer, X As Single, Y As Single)

    If pulsante And LEFT_BUTTON Then

        SavePicture Immagine1.Image, "c:\project\annulla.bmp"
        Immagine1.CurrentX = X
        x1 = X
        Immagine1.CurrentY = Y
        y1 = Y
        Label4.Caption = "0"
        Label5.Caption = "0"
```

```
        Else

                Immagine1.Cls
                Immagine1.Picture = LoadPicture("c:\project\bright.dib")
                Immagine1.CurrentX = X
                Immagine1.CurrentY = Y

        End If

End Sub

Private Sub Immagine1_MouseMove(Botton As Integer, Shift As Integer, X As Single, Y As Single)

        label2.Caption = "บริเวณนี้วาดภาพได้"
        Label1.Caption = X
        Label3.Caption = Y


        If Botton And LEFT_BUTTON Then
                Immagine1.Line -(X, Y)
                x2 = X
                y2 = Y
                d = ((x2 - x1) ^ 2 + (y2 - y1) ^ 2) ^ (1 / 2)
                e = 3.1416 * d
              Label4.Caption = d
              Label5.Caption = e
        End If


         If pulsante And LEFT_BUTTON And (Botton And RIGHT_BUTTON) Then
                SavePicture Immagine1.Image, "c:\project\undomo.bmp"
                Immagine1.Cls
                Immagine1.Picture = LoadPicture("c:project\bright.dib")
        End If

End Sub


Private Sub MnuBincont_Click()
    Unload Prepare
    Load BinContForm
    BinContForm.Show

End Sub



Private Sub mnuexit_Click()
    End
```

```
End Sub

Private Sub mnuFileLoad_Click()
    Load Loadfile
    Loadfile.Show
End Sub

Private Sub mnuFilter_Click()
        Unload Prepare
        Load FilterForm
        FilterForm.Show
End Sub

Private Sub mnusalva_Click()

    Load Save
    Save.Show 1

End Sub


Private Sub mnusave_Click()
    Load Save
    Save.Show 1
End Sub


Private Sub Option1_Click()

        Immagine1.DrawMode = 1
        Immagine1.DrawWidth = 1
        Immagine1.MousePointer = 2

End Sub

Private Sub Option1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

        label2.Caption = "ถ้าเลือกปุ่มนี้จะวาดภาพ"

End Sub

Private Sub Option2_Click()

        Immagine1.DrawMode = 16
        Immagine1.DrawWidth = 12
        Immagine1.MousePointer = 4

End Sub
```

```
Private Sub Option2_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

    label2.Caption = "ถ้าเลือกปุ่มนี้จะลบภาพ"

End Sub



Private Sub Option3_Click()

    Immagine1.DrawMode = 3
    Immagine1.DrawWidth = 1
    Immagine1.MousePointer = 2

End Sub

Private Sub Option3_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

    label2.Caption = "ถ้าเลือกปุ่มนี้จะวัดเส้นผ่าศูนย์กลาง"

End Sub
```

ไฟล์ load.frm

```
VERSION 4.00
Begin VB.Form Loadfile
   Appearance      =  0  'Flat
   BackColor       =  &H80000005&
   Caption         =  "Load a file"
   ClientHeight    =  4260
   ClientLeft      =  1890
   ClientTop       =  1485
   ClientWidth     =  4530
   BeginProperty Font
      name            =  "MS Sans Serif"
      charset         =  1
      weight          =  700
      size            =  8.25
      underline       =  0   'False
      italic          =  0   'False
      strikethrough   =  0   'False
   EndProperty
   ForeColor       =  &H80000008&
   Height          =  4665
   Left            =  1830
   LinkMode        =  1  'Source
   LinkTopic       =  "Form1"
   ScaleHeight     =  540
   ScaleWidth      =  540
   Top             =  1140
   Width           =  4650
   Begin VB.TextBox fname
      Appearance      =  0  'Flat
      Height          =  375
      Left            =  240
      TabIndex        =  4
      Top             =  2880
      Width           =  4215
   End
   Begin VB.CommandButton Ok
      Appearance      =  0  'Flat
      BackColor       =  &H80000005&
      Caption         =  'Ok"
      Height          =  492
      Left            =  1680
      TabIndex        =  3
      Top             =  3600
      Width           =  1332
   End
   Begin VB.DriveListBox Drive
      Appearance      =  0  'Flat
```

```
    Height          =   300
    Left            =   2520
    TabIndex        =   2
    Top             =   2400
    Width           =   1932
  End
  Begin VB.DirListBox Directory
    Appearance      =   0  'Flat
    Height          =   2052
    Left            =   2520
    TabIndex        =   1
    Top             =   240
    Width           =   1932
  End
  Begin VB.FileListBox File
    Appearance      =   0  'Flat
    Height          =   2370
    Left            =   240
    TabIndex        =   0
    Top             =   240
    Width           =   1935
  End
End
Attribute VB_Name = "Loadfile"
Attribute VB_Creatable = False
Attribute VB_Exposed = False


Private Sub Directory_Change()

    file.filename = Directory.Path
    fname = file.Path

End Sub


Private Sub Directory_KeyDown(keycode As Integer, Shift As Integer)

Const KEY_RETURN = &HD

If keycode = KEY_RETURN Then

        Prepare.Immagine1.Picture = LoadPicture(fname)
        Unload Loadfile

        Exit Sub
End If

End Sub
```

```
Private Sub Drive_Change()

        On Error GoTo dnr
        Directory.Path = Drive.Drive
        Exit Sub

dnr: MsgBox ("drive not ready")
        Exit Sub

End Sub



Private Sub Drive_KeyDown(keycode As Integer, Shift As Integer)

Const KEY_RETURN = &HD

If keycode = KEY_RETURN Then
        Prepare.Immagine1.Picture = LoadPicture(fname)
        Unload Loadfile
        Exit Sub
End If

End Sub


Private Sub fname_KeyDown(keycode As Integer, Shift As Integer)

Const KEY_RETURN = &HD

If keycode = KEY_RETURN Then
        Prepare.Immagine1.Picture = LoadPicture(fname)
        Unload Loadfile
        Exit Sub
End If

End Sub


Private Sub File_Click()

If file.Path = "c:\" Or file.Path = "a:\" Then
    fname = file.Path + file.filename
Else
    fname = file.Path + "\" + file.filename
End If
```

```
End Sub


Private Sub File_KeyDown(keycode As Integer, Shift As Integer)

Const KEY_RETURN = &HD

If keycode = KEY_RETURN Then
        Prepare.Immagine1.Picture = LoadPicture(fname)
        Unload Loadfile
        Exit Sub
End If

End Sub


Private Sub File_PathChange()

   fname = file.Path + "\"
End Sub



Private Sub Form_KeyDown(keycode As Integer, Shift As Integer)

Const KEY_RETURN = &HD

If keycode = KEY_RETURN Then
        Prepare.Immagine1.Picture = LoadPicture(fname)
        Unload Loadfile
        Exit Sub
End If

End Sub


Private Sub ok_click()

        On Error GoTo fner

        Prepare.Immagine1.Picture = LoadPicture(fname)

        Unload Loadfile
        Exit Sub

fner: MsgBox ("The name is not valid")
        Exit Sub

End Sub
```

ไฟล์ count.frm

```
VERSION 4.00
Begin VB.Form Coun
   Appearance      =  0  'Flat
   BackColor       =  &H00C0C0C0&
   Caption         =  "ภาควิชานิวเคลียร์เทคโนโลยี"
   ClientHeight    =  6780
   ClientLeft      =  390
   ClientTop       =  465
   ClientWidth     =  8310
   BeginProperty Font
      name            =  "MS Sans Serif"
      charset         =  1
      weight          =  700
      size            =  8.25
      underline       =  0  'False
      italic          =  0  'False
      strikethrough   =  0  'False
   EndProperty
   ForeColor       =  &H00000000&
   Height          =  7185
   Left            =  330
   LinkTopic       =  "Form1"
   ScaleHeight     =  6780
   ScaleWidth      =  8310
   Top             =  120
   Width           =  8430
   WindowState     =  2  'Maximized
   Begin VB.CommandButton Command2
      Appearance      =  0  'Flat
      BackColor       =  &H80000005&
      Caption         =  "กลับไปดูรูปเดิม"
      Height          =  975
      Left            =  6840
      TabIndex        =  1
      Top             =  5280
      Width           =  1695
   End
   Begin VB.PictureBox Immagine1
      Appearance      =  0  'Flat
      AutoRedraw      =  -1  'True
      BackColor       =  &H00FFFFFF&
      DrawMode        =  1  'Blackness
      DrawWidth       =  4
      FillStyle       =  0  'Solid
      ForeColor       =  &H80000008&
      Height          =  5310
      Left            =  1200
```

```
    ScaleHeight    =   352
    ScaleMode      =   0  'User
    ScaleWidth     =   352
    TabIndex       =   0
    Top            =   960
    Width          =   5310
  End
End
Attribute VB_Name = "Coun"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Dim Xorig, Yorig As Integer
Dim Xcour, Ycour As Integer
Private Declare Function preparefile Lib "C:\project\Dll1.dll" (ByVal i As Integer) As Integer



Private Sub Command2_Click()

   Unload Coun

End Sub


Private Sub Form_Load()

Dim a, b, c, d, ex As Integer
Dim si  As Integer
Dim cont As Integer
Dim avar, linea, linea1, size As String
Dim p, s, l, sp, ss, sl As Integer

   size$ = "Enter number of pixel to classify."
   Title$ = "Input Size Pixel"
   Defalt$ = "100"
   si = val(InputBox$(size$, Title$, Defalt$))
   d = 0
   sp = 0
   ss = 0
   sl = 0

   Do While ex <> 1

      Immagine1.Picture = LoadPicture("c:\project\px2.bmp")
      Screen.MousePointer = 11

      i = preparefile(1)
```

```
Screen.MousePointer = 0

Rep$ = Dir$("c:\*.*")
Do While Rep$ <> ""
Rep$ = Dir$
Loop


Coun.Immagine1.DrawWidth = 8
Coun.Immagine1.DrawMode = 16
Open "c:\project\numero1.txt" For Input As #1

Line Input #1, linea
cont = val(linea)
n = 1
Do While Not EOF(1)
      Line Input #1, linea
      Line Input #1, linea1
      a = val(linea)
      b = val(linea1)
      Coun.Immagine1.PSet (b, 352 - a)
      n = n + 1
Loop
Close #1
 SavePicture Immagine1.Image, "c:\project\px2.bmp"

 If n = 1 Then ex = 1
 If n > si Then l = l + 1

d = d + 1

Loop

   msg1 = "Total Object = " + Str$(d - 1)
   msg2 = "  large = " + Str$(l)
   msg3 = "  small = " + Str$(d - l - 1)
   MsgBox msg1 + msg2 + msg3

End Sub
```

ไฟล์ readbmp.c

```
/*****************************************************************/
/*                      READBITMAP                               */
/*****************************************************************/
/*      Reads the bitmap matrix of the input picture.            */
/*      Returns a matrix with 0 and 1.                           */
/*****************************************************************/
/*DATA:                                                          */
/* mat_in:  matrix which contains the input picture.             */
/*                      (Output)                                 */
/*****************************************************************/

#include <stdlib.h>
#include <windows.h>
#include <stdio.h>
#include "varglob.h"

/*****************************************************************/
/*                      variables of routine READBITMAP          */
/*****************************************************************/

BITMAPINFO1  *pbm;          /*      information on the size and the
                                    colours of the bitmap           */
int numbits;                /*      size of the picture bitmap      */
int numquads = 0;           /*      number of the colours of the bitmap */
BITMAPFILEHEADER1 fbm;      /*      bitmap file header              */
BITMAPINFOHEADER1 bmi;      /*      bitmap information header       */


int far pascal  readbitmap()
{
        FILE        *fb;            /*      binary file             */
        int         i,j;            /*      locating counter of the matrix  */
        int         dimpx,dimpy;    /*      size of the bitmap      */
        unsigned char       elm;    /*      elements of the bitmap  */


        /*      open the file px2.bmp which contains the picture    */
        if((fb = fopen("c:..\\project\\px2.bmp","rb")) == NULL)
                DEMO_BADFILE1;

        /*      verify that the header of the file isn't empty      */
        if(fread(&fbm, sizeof(fbm),1,fb) != 1)
                DEMO_BADFILE1;

        /*      verify that the type of file is right: 0x4D42       */
        if(fbm.bfType != 0x4d42) DEMO_BADFILE1;
```

```
            /*          preset to "numbits" the real size of matrix bitmap          */
            numbits = fbm.bfSize - fbm.bfOffBits;

            /*          verify that the header of the bitmap isn't empty            */
             if(fread(&bmi, sizeof(bmi),1,fb) != 1) DEMO_BADFILE1;

             /*         preset the size of bitmap                                   */
   dimpx = bmi.biWidth;
             if(dimpx != DIMMAT_IN) DEMO_BADFILE1;
   dimpy = bmi.biHeight;
             if(dimpy != DIMMAT_IN) DEMO_BADFILE1;

/*           if the number of colours (biClrUsed) is 0 then the numbers          */
/*           of colours that must be read (numquads) is 1, shift of biBitCount,  */
/*           otherwise preset the number of colours                              */

   if(bmi.biClrUsed == 0) numquads = 1 << bmi.biBitCount;
             else numquads = bmi.biClrUsed;

/*           set to the beginning of the matrix bitmap                           */
             fseek(fb, fbm.bfOffBits, SEEK_SET);

             /*         version 256 colours       */
                 if (bmi.biBitCount == 8)
              {
                    for (i=0;i<DIMMAT_IN;i++)
                     for (j=0;j<DIMMAT_IN;j++) {
                              if(fread(&elm,1,1,fb)!=1){
                              DEMO_BADFILE1
                               }
                              mat_in[i][j]=(elm && 19 ? 0:1);
                              }
    }

             /*         version 16 colours        */
             if (bmi.biBitCount==4)
   {
             for (i=0;i<DIMMAT_IN;i++)
                     for (j=0;j<DIMMAT_IN;j+=2) {
                              if(fread(&elm,1,1,fb) != 1)
                                       DEMO_BADFILE1;
                                                if (elm ==255){
                                                         mat_in[i][j] = 0;
                                                         mat_in[i][j+1] = 0;
                                                         }
                                                else if (elm == 0){
                                                         mat_in[i][j] = 1;
                                                         mat_in[i][j+1] = 1;
                                                         }
```

```
                                        else if (elm == 15){
                                                mat_in[i][j] = 1;
                                                mat_in[i][j+1] = 0;
                                                }
                                                else if (elm == 240){
                                                mat_in[i][j] = 0;
                                                mat_in[i][j+1] = 1;

                                                }
                                        }
                                }
                /*      surround the picture of zeroes  */
                  for (i=0;i<DIMMAT_IN;i++)
                  {
                        mat_in[i][0] = 0;
                        mat_in[i][DIMMAT_IN-1] = 0;
                        mat_in[0][i] = 0;
                        mat_in[DIMMAT_IN-1][i] = 0;
                  }

        fclose(fb);

return (1);
}
```

ไฟล์ edge.c

```
/*************************************************************************/
/*                                EDGE                                   */
/*************************************************************************/
/*      Once taken in input the matrix mat_in that contains the image, the */
/*      EDGE procedure extracts the contour of it.                        */
/*       The procedure reads the matrix mat_in until it meets the first black */
/*      pixel, then it follows the edge until it hits again the first met pixel. */
/*      In case the points of the edge exceed the available memory the routine */
/*      exits and signals the event.                                      */
/*       The subroutine take_data writes the values of                    */
/*      the abscissae and of the ordinates of the edge on the file numero1.txt. */
/*      Finally the contour is sampled (1 point on 9).                    */
/*************************************************************************/
/* DATA:                                                                 */
/* mat_in:   matrix that contains the image in input.          (Input)   */
/* bordo.asc: array that contains the values of the abscissae of the     */
/*                  contour of the image.                      (Output)  */
/*bordo.ord: array that contains the values of the ordinates of the      */
/*                  contour of the image.                      (Output)  */
/* cont:             number of points of the contour of the image.       */
/*                  (Output)                                             */
/*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "varglob.h"

/*********************************************************************/
/*                  list of subroutine                               */
/*********************************************************************/

void def_edge(int,int);
void take_data();

/*-------------------------------------------------------------------*/
/*      take_data:  edge's subroutine                                */
/*-------------------------------------------------------------------*/

void take_data ( void){
        FILE *fp;
        int j;  /*   locating counter          */

/*      opens the file data.txt     */
fp=fopen("c:..\\project\\data.txt","w");

/*      writes the number of points of the edge          */
```

```
fprintf(fp,"%d\n",cont);
for(j=1;j<=cont;j++){
/*          writes the values of the ordinates of the points of the edge      */
                    fprintf(fp,"%d",bordo[j].ord);
                              fprintf(fp,"\n");
/* writes the values of the abscissae of the points of the edge    */
      fprintf(fp,"%d",bordo[j].asc);
                              fprintf(fp,"\n");
          }
/*          closes the file data.txt    */
fclose(fp);
return ; /* take_data */
}

/*-------------------------------------------------------*/
/* def_edge: edge's subroutine                           */
/*-------------------------------------------------------*/

void def_edge(h,k)
int h,k;
{
                    bordo[cont].asc=k;
                    bordo[cont].ord=h;
                    cont++;
                    /*          checks the number of the points of the edge          */
                    if(cont>=MAX_PUNTI)
                              DEMO_BADFILE2;
                    return ;
 }    /* def_edge */
/*-------------------------------------------------------*/

int far pascal edge(){
int i,j;      /*          locating counters of the columns and of the rows    */
int i1,i0,j1,j0;
int k;
cont=1;

/*          reads the matrix mat_in and loads the array of the abscissas and of the ordinates  of the edge          */
 for (i=0;i<DIMMAT_IN;i++){
          for (j=0;j<DIMMAT_IN;j++)
           /*          searches the first black point of the  matrix          */
                              if (mat_in[i][j]==1){
                               def_edge(i,j);
                                break;
                               }
                              if (cont==2)
                               break;
                              }
```

```
/*          cycle for the search of the second point of the edge          */
                    j=bordo[1].asc;
                    i=bordo[1].ord;
                    if (mat_in[i][j-1]==1)
                    def_edge(i,j-1);
                   else if (mat_in[i-1][j-1]==1)
                              def_edge(i-1,j-1);
                              else if (mat_in[i-1][j]==1)
                                        def_edge(i-1,j);
                                         else if (mat_in[i-1][j+1]==1)
                                                  def_edge(i-1,j+1);
                                                  else if (mat_in[i][j+1]==1)
                                                            def_edge(i,j+1);
                                                            else if (mat_in[i+1][j+1]==1)
                                                                      def_edge(i+1,j+1);
                                                                      else if (mat_in[i+1][j]==1)
                                                                                def_edge(i+1,j);
                                                                                else if (mat_in[i+1][j-1]==1)
                                                                                  def_edge(i+1,j-1);

/*          cycle for the search of the others point of the edge          */
    k=2;
    do{
        j1=bordo[k].asc;
        i1=bordo[k].ord;
        j0=bordo[k-1].asc;
        i0=bordo[k-1].ord;
        /*          control on the neighbours of the same row          */
        if (i0==i1){
            /*      right or left neighbour ?   */
            if (j0<j1){
                if (mat_in[i1-1][j1-1]==1)
                    def_edge(i1-1,j1-1);
                    else if(mat_in[i1-1][j1])
                        def_edge(i1-1,j1);
                        else if(mat_in[i1-1][j1+1])
                            def_edge(i1-1,j1+1);
                            else if(mat_in[i1][j1+1])
                                def_edge(i1,j1+1);
                                else if(mat_in[i1+1][j1+1])
                                    def_edge(i1+1,j1+1);
                                    else if(mat_in[i1+1][j1])
                                        def_edge(i1+1,j1);
                                        else if(mat_in[i1+1][j1-1])
                                            def_edge(i1+1,j1-1);
                                            else
                                              def_edge(i0,j0);
            }/* if */
            else{
```

```
            if (mat_in[i1+1][j1+1]==1)
                def_edge(i1+1,j1+1);
                else if(mat_in[i1+1][j1])
                    def_edge(i1+1,j1);
                    else if(mat_in[i1+1][j1-1])
                        def_edge(i1+1,j1-1);
                        else if(mat_in[i1][j1-1])
                            def_edge(i1,j1-1);
                            else if(mat_in[i1-1][j1-1])
                                def_edge(i1-1,j1-1);
                                else if(mat_in[i1-1][j1])
                                    def_edge(i1-1,j1);
                                    else if(mat_in[i1-1][j1+1])
                                        def_edge(i1-1,j1+1);
                                        else
                                          def_edge(i0,j0);
        }/* else */
} /*        first "if"      */
/*          control on the neighbours of the same column        */
else if (j0==j1){
        /*      higher or lower neighbour ?             */
        if (i0<i1){
            if(mat_in[i1-1][j1+1]==1)
                def_edge(i1-1,j1+1);
                else if(mat_in[i1][j1+1])
                    def_edge(i1,j1+1);
                    else if(mat_in[i1+1][j1+1])
                        def_edge(i1+1,j1+1);
                        else if(mat_in[i1+1][j1])
                            def_edge(i1+1,j1);
                            else if(mat_in[i1+1][j1-1])
                                def_edge(i1+1,j1-1);
                                else if(mat_in[i1][j1-1])
                                    def_edge(i1,j1-1);
                                    else if(mat_in[i1-1][j1-1])
                                        def_edge(i1-1,j1-1);
                                        else
                                          def_edge(i0,j0);
        }/* if*/
        else{
            if(mat_in[i1+1][j1-1]==1)
                def_edge(i1+1,j1-1);
                else if(mat_in[i1][j1-1])
                    def_edge(i1,j1-1);
                    else if(mat_in[i1-1][j1-1])
                        def_edge(i1-1,j1-1);
                        else if(mat_in[i1-1][j1])
                            def_edge(i1-1,j1);
                            else if(mat_in[i1-1][j1+1])
```

```
                                        def_edge(i1-1,j1+1);
                                        else if(mat_in[i1][j1+1])
                                            def_edge(i1,j1+1);
                                            else if(mat_in[i1+1][j1+1])
                                                def_edge(i1+1,j1+1);
                                                else
                                                  def_edge(i0,j0);
            }/* else */
        }/*         second "if"   */
                /*          control on the neighbours diagonally to the left      */
        else if(j0<j1){
                /* higher or lower neighbour ?              */
if(i0<i1){
                if(mat_in[i1-1][j1]==1)
                    def_edge(i1-1,j1);
                    else if(mat_in[i1-1][j1+1])
                        def_edge(i1-1,j1+1);
                        else if(mat_in[i1][j1+1])
                            def_edge(i1,j1+1);
                            else if(mat_in[i1+1][j1+1])
                                def_edge(i1+1,j1+1);
                                else if(mat_in[i1+1][j1])
                                    def_edge(i1+1,j1);
                                    else if(mat_in[i1+1][j1-1])
                                        def_edge(i1+1,j1-1);
                                        else if(mat_in[i1][j1-1])
                                            def_edge(i1,j1-1);
                                            else
                                              def_edge(i0,j0);
            }/* if */
            else{
                if(mat_in[i1][j1-1]==1)
                    def_edge(i1,j1-1);
                    else if(mat_in[i1-1][j1-1])
                        def_edge(i1-1,j1-1);
                        else if(mat_in[i1-1][j1])
                            def_edge(i1-1,j1);
                            else if(mat_in[i1-1][j1+1])
                                def_edge(i1-1,j1+1);
                                else if(mat_in[i1][j1+1])
                                    def_edge(i1,j1+1);
                                    else if(mat_in[i1+1][j1+1])
                                        def_edge(i1+1,j1+1);
                                        else if(mat_in[i1+1][j1])
                                            def_edge(i1+1,j1);
                                            else
                                              def_edge(i0,j0);
            }/* else */
        }/*         third "if"     */
```

```
                    /*          control on the neighbours diagonally to the right     */
else{
                /*      higher or lower neighbour ?             */
if(i0<i1){
                    if(mat_in[i1][j1+1]==1)
                        def_edge(i1,j1+1);
                        else if(mat_in[i1+1][j1+1])
                            def_edge(i1+1,j1+1);
                            else if(mat_in[i1+1][j1])
                                def_edge(i1+1,j1);
                                else if(mat_in[i1+1][j1-1])
                                    def_edge(i1+1,j1-1);
                                    else if(mat_in[i1][j1-1])
                                        def_edge(i1,j1-1);
                                        else if(mat_in[i1-1][j1-1])
                                            def_edge(i1-1,j1-1);
                                            else if(mat_in[i1-1][j1])
                                                def_edge(i1-1,j1);
                                                else
                                                    def_edge(i0,j0);
            }/* if */
            else{
                if(mat_in[i1+1][j1]==1)
                    def_edge(i1+1,j1);
                    else if(mat_in[i1+1][j1-1])
                        def_edge(i1+1,j1-1);
                        else if(mat_in[i1][j1-1])
                            def_edge(i1,j1-1);
                            else if(mat_in[i1-1][j1-1])
                                def_edge(i1-1,j1-1);
                                else if(mat_in[i1-1][j1])
                                    def_edge(i1-1,j1);
                                    else if(mat_in[i1-1][j1+1])
                                        def_edge(i1-1,j1+1);
                                        else if(mat_in[i1][j1+1])
                                            def_edge(i1,j1+1);
                                            else
                                                def_edge(i0,j0);
            }/* else */
        }/*         fourth "if"     */
                                                k++;
                /*          until it reaches the first met pixel        */
        } while ((bordo[cont-1].asc!=bordo[1].asc) || (bordo[cont-1].ord!=bordo[1].ord));
        cont=cont-2;

take_data();

k=0;
```

```
/*              samples the edge ( 1 point on 9)        */
for (j=0;j<cont;j++)
                        if ((j % 9)==0){
            i=(j/9)+1;
            bordo[i].asc=bordo[j+1].asc;
            bordo[i].ord=bordo[j+1].ord;
            k++;
            }

cont=k; /*   number of the points of the sampled edge            */

return(0) ;

}
```

ไฟล์ global.bas

```
Attribute VB_Name = "Module1"

Global topview As Integer
Global count1  As Integer

Global Const VRAI = True
Global Const FAUX = False

Global Const CF_LINK = &HBF00
Global Const CF_TEXT = 1
Global Const CF_BITMAP = 2
Global Const CF_METAFILE = 3
Global Const CF_DIB = 8
Global Const CF_PALETTE = 9


Global Const ENTER = 0
Global Const LEAVE = 1
Global Const OVER = 2


Global Const CANCEL = 0
Global Const BEGIN_DRAG = 1
Global Const END_DRAG = 2

Global Const MODAL = 1
Global Const MODELESS = 0

Global Const TILE_HORIZONTAL = 1
Global Const TILE_VERTICAL = 2
Global Const ARRANGE_ICONS = 3

Global Const BRINGTOFRONT = 0
Global Const SENDTOBACK = 1

Global Const KEY_LBUTTON = &H1
Global Const KEY_RBUTTON = &H2
Global Const KEY_CANCEL = &H3
Global Const KEY_MBUTTON = &H4
Global Const KEY_BACK = &H8
Global Const KEY_TAB = &H9
Global Const KEY_CLEAR = &HC
Global Const KEY_RETURN = &HD
Global Const KEY_SHIFT = &H10
Global Const KEY_CONTROL = &H11
Global Const KEY_MENU = &H12
Global Const KEY_PAUSE = &H13
```

```
Global Const KEY_CAPITAL = &H14
Global Const KEY_ESCAPE = &H1B
Global Const KEY_SPACE = &H20
Global Const KEY_PRIOR = &H21
Global Const KEY_NEXT = &H22
Global Const KEY_END = &H23
Global Const KEY_HOME = &H24
Global Const KEY_LEFT = &H25
Global Const KEY_UP = &H26
Global Const KEY_RIGHT = &H27
Global Const KEY_DOWN = &H28
Global Const KEY_SELECT = &H29
Global Const KEY_PRINT = &H2A
Global Const KEY_EXECUTE = &H2B
Global Const KEY_SNAPSHOT = &H2C
Global Const KEY_INSERT = &H2D
Global Const KEY_DELETE = &H2E
Global Const KEY_HELP = &H2F


Global Const KEY_NUMPAD0 = &H60
Global Const KEY_NUMPAD1 = &H61
Global Const KEY_NUMPAD2 = &H62
Global Const KEY_NUMPAD3 = &H63
Global Const KEY_NUMPAD4 = &H64
Global Const KEY_NUMPAD5 = &H65
Global Const KEY_NUMPAD6 = &H66
Global Const KEY_NUMPAD7 = &H67
Global Const KEY_NUMPAD8 = &H68
Global Const KEY_NUMPAD9 = &H69
Global Const KEY_MULTIPLY = &H6A
Global Const KEY_ADD = &H6B
Global Const KEY_SEPARATOR = &H6C
Global Const KEY_SUBTRACT = &H6D
Global Const KEY_DECIMAL = &H6E
Global Const KEY_DIVIDE = &H6F
Global Const KEY_F1 = &H70
Global Const KEY_F2 = &H71
Global Const KEY_F3 = &H72
Global Const KEY_F4 = &H73
Global Const KEY_F5 = &H74
Global Const KEY_F6 = &H75
Global Const KEY_F7 = &H76
Global Const KEY_F8 = &H77
Global Const KEY_F9 = &H78
Global Const KEY_F10 = &H79
Global Const KEY_F11 = &H7A
Global Const KEY_F12 = &H7B
Global Const KEY_F13 = &H7C
```

```
Global Const KEY_F14 = &H7D
Global Const KEY_F15 = &H7E
Global Const KEY_F16 = &H7F

Global Const KEY_NUMLOCK = &H90

Global Const V_EMPTY = 0
Global Const V_NULL = 1
Global Const V_INTEGER = 2
Global Const V_LONG = 3

Global Const V_SINGLE = 4
Global Const V_DOUBLE = 5
Global Const V_CURRENCY = 6
Global Const V_DATE = 7
Global Const V_STRING = 8

Global Const WRONG_FORMAT = 1
Global Const DDE_SOURCE_CLOSED = 6
Global Const TOO_MANY_LINKS = 7
Global Const DATA_TRANSFER_FAILED = 8

Global Const FORM_CONTROLMENU = 0
Global Const FORM_CODE = 1

Global Const APP_TASKMANAGER = 3
Global Const FORM_MDIFORM = 4

Global Const BLACK = &H0&
Global Const RED = &HFF&
Global Const GREEN = &HFF00&
Global Const YELLOW = &HFFFF&
Global Const BLUE = &HFF0000
Global Const WHITE = &HFFFFFF

Global Const SCROLL_BARS = &H80000000
Global Const DESKTOP = &H80000001
Global Const ACTIVE_TITLE_BAR = &H80000002
Global Const INACTIVE_TITLE_BAR = &H80000003
Global Const MENU_BAR = &H80000004
Global Const WINDOW_BACKGROUND = &H80000005
Global Const WINDOW_FRAME = &H80000006
Global Const MENU_TEXT = &H80000007
Global Const WINDOW_TEXT = &H80000008
Global Const TITLE_BAR_TEXT = &H80000009

Global Const ACTIVE_BORDER = &H8000000A
Global Const INACTIVE_BORDER = &H8000000B
Global Const APPLICATION_WORKSPACE = &H8000000C
```

```
Global Const HIGHLIGHT = &H8000000D
Global Const HIGHLIGHT_TEXT = &H8000000E
Global Const BUTTON_FACE = &H8000000F
Global Const BUTTON_SHADOW = &H80000010
Global Const GRAY_TEXT = &H80000011


Global Const BUTTON_TEXT = &H80000012

Global Const NONE = 0
Global Const ALIGN_TOP = 1
Global Const ALIGN_BOTTOM = 2


Global Const LEFT_JUSTIFY = 0
Global Const RIGHT_JUSTIFY = 1
Global Const CENTER = 2

Global Const TRAIT_SIMPLE_FIXE = 1
Global Const REGLABLE = 2
Global Const TRAIT_DOUBLE_FIXE = 3

Global Const FIXED_SINGLE = 1
Global Const SIZABLE = 2
Global Const FIXED_DOUBLE = 3

Global Const DEFAULT = 0
Global Const ARROW = 1
Global Const CROSSHAIR = 2
Global Const IBEAM = 3
Global Const ICON_POINTER = 4
Global Const SIZE_POINTER = 5
Global Const SIZE_NE_SW = 6
Global Const SIZE_N_S = 7
Global Const SIZE_NW_SE = 8
Global Const SIZE_W_E = 9
Global Const HOURGLASS = 11
Global Const NO_DROP = 12


Global Const MANUAL = 0
Global Const AUTOMATIC = 1


Global Const BLACKNESS = 1
Global Const NOT_MERGE_PEN = 2
Global Const MASK_NOT_PEN = 3
```

```
Global Const NOT_COPY_PEN = 4
Global Const MASK_PEN_NOT = 5
Global Const INVERT = 6
Global Const XOR_PEN = 7
Global Const NOT_MASK_PEN = 8
Global Const MASK_PEN = 9
Global Const NOT_XOR_PEN = 10
Global Const NOP = 11
Global Const MERGE_NOT_PEN = 12
Global Const COPY_PEN = 13
Global Const MERGE_PEN_NOT = 14
Global Const MERGE_PEN = 15
Global Const WHITENESS = 16


Global Const SOLID = 0
Global Const DASH = 1
Global Const DOT = 2
Global Const DASH_DOT_DOT = 4
Global Const INVISIBLE = 5
Global Const INSIDE_SOLID = 6

Global Const TRANSPARENT = 1
Global Const HORIZONTAL_LINE = 2
Global Const VERTICAL_LINE = 3
Global Const UPWARD_DIAGONAL = 4
Global Const DOWNWARD_DIAGONAL = 5
Global Const CROSS = 6
Global Const DIAGONAL_CROSS = 7



Global Const AUTO = 1
Global Const SOURCE = 1
Global Const SURDEMANDE = 2

Global Const HOT = 1
Global Const SERVER = 1
Global Const COLD = 2


Global Const LINK_SOURCE = 1
Global Const LINK_AUTOMATIC = 1
Global Const LINK_MANUAL = 2
Global Const LINK_NOTIFY = 3

Global Const USER = 0
Global Const TWIPS = 1
Global Const POINTS = 2
```

```
Global Const PIXELS = 3
Global Const CHARACTERS = 4
Global Const INCHES = 5
Global Const MILLIMETERS = 6
Global Const CENTIMETERS = 7

Global Const HORIZONTALE = 1
Global Const VERTICALE = 2
Global Const COMBINE = 3

Global Const HORIZONTAL = 1
Global Const VERTICAL = 2
Global Const BOTH = 3

Global Const SHAPE_RECTANGLE = 0
Global Const SHAPE_SQUARE = 1
Global Const SHAPE_OVAL = 2
Global Const SHAPE_CIRCLE = 3
Global Const SHAPE_ROUNDED_RECTANGLE = 4
Global Const SHAPE_ROUNDED_SQUARE = 5

Global Const NORMAL = 0
Global Const MINIMIZED = 1
Global Const MAXIMIZED = 2

Global Const UNCHECKED = 0
Global Const CHECKED = 1
Global Const GRAYED = 2

Global Const SHIFT_MASK = 1
Global Const CTRL_MASK = 2
Global Const ALT_MASK = 4

Global Const LEFT_BUTTON = 1
Global Const RIGHT_BUTTON = 2
Global Const MIDDLE_BUTTON = 4

Global Const MB_OKCANCEL = 1
Global Const MB_ABORTRETRYIGNORE = 2
Global Const MB_YESNOCANCEL = 3
Global Const MB_YESNO = 4
Global Const MB_RETRYCANCEL = 5

Global Const MB_ICONSTOP = 16
Global Const MB_ICONQUESTION = 32
Global Const MB_ICONEXPLICATION = 48
Global Const MB_ICONEXCLAMATION = 48
Global Const MB_ICONINFORMATION = 64
```

```
Global Const MB_APPLMODAL = 0
Global Const MB_DEFBUTTON1 = 0
Global Const MB_DEFBUTTON2 = 256
Global Const MB_DEFBUTTON3 = 512
Global Const MB_SYSTEMMODAL = 4096


Global Const IDOK = 1
Global Const IDCANCEL = 2
Global Const IDABORT = 3
Global Const IDRETRY = 4
Global Const IDIGNORE = 5
Global Const IDYES = 6
Global Const IDNO = 7


Global Const ATTR_NORMAL = 0
Global Const ATTR_READONLY = 1
Global Const ATTR_HIDDEN = 2
Global Const ATTR_SYSTEM = 4
Global Const ATTR_VOLUME = 8
Global Const ATTR_DIRECTORY = 16
Global Const ATTR_ARCHIVE = 32

Global Const GRID_ALIGNLEFT = 0
Global Const GRID_ALIGNRIGHT = 1
Global Const GRID_ALIGNCENTER = 2

Global Const GRID_SIMPLE = 0
Global Const GRID_REPETITION = 1

Global Const GRID_SINGLE = 0
Global Const GRID_REPEAT = 1

Global Const OLE_CREATE_NEW = 0
Global Const OLE_CREATE_FROM_FILE = 1
Global Const OLE_COPY = 4
Global Const OLE_PASTE = 5
Global Const OLE_UPDATE = 6
Global Const OLE_ACTIVATE = 7
Global Const OLE_EXECUTE = 8
Global Const OLE_CLOSE = 9
Global Const OLE_DELETE = 10
Global Const OLE_SAVE_TO_FILE = 11
Global Const OLE_READ_FROM_FILE = 12
Global Const OLE_CONVERT_TO_TYPE = 13


Global Const OLE_LIE = 0
```

```
Global Const OLE_INCORPORE = 1
Global Const OLE_STATIQUE = 2

Global Const OLE_LINKED = 0
Global Const OLE_EMBEDDED = 1
Global Const OLE_STATIC = 2


Global Const OLE_MODIFIE = 0
Global Const OLE_ENREGISTRE = 1
Global Const OLE_FERME = 2
Global Const OLE_LIBERE = 3

Global Const OLE_CHANGED = 0
Global Const OLE_SAVED = 1
Global Const OLE_CLOSED = 2
Global Const OLE_RELEASE = 3

'   subfunction that  presets tha value 1 to the global
'   constant "count1" when the button "Command2" of the form
'   "Prepare.frm" is pressed
'
Sub prova()

    count1 = 1

End Sub
```

ไฟล์ form1.frm

```
VERSION 4.00
Begin VB.Form Form1
   Caption         =  "ภาควิชานิวเคลียร์เทคโนโลยี"
   ClientHeight    =  3000
   ClientLeft      =  1470
   ClientTop       =  1515
   ClientWidth     =  6825
   Height          =  3405
   Left            =  1410
   LinkTopic       =  "Form1"
   ScaleHeight     =  3000
   ScaleWidth      =  6825
   Top             =  1170
   Width           =  6945
   Begin VB.Label Label3
      Caption         =  "อภิชาติ สระวาสี"
      BeginProperty Font
         name            =  "EucrosiaUPC"
         charset         =  1
         weight          =  700
         size            =  14.25
         underline       =  0   'False
         italic          =  0   'False
         strikethrough   =  0   'False
      EndProperty
      Height          =  375
      Left            =  2760
      TabIndex        =  2
      Top             =  1560
      Width           =  1335
   End
   Begin VB.Label Label2
      Caption         =  "โดย"
      BeginProperty Font
         name            =  "MS Sans Serif"
         charset         =  1
         weight          =  700
         size            =  9.75
         underline       =  0   'False
         italic          =  0   'False
         strikethrough   =  0   'False
      EndProperty
      Height          =  375
      Left            =  3120
      TabIndex        =  1
      Top             =  1080
      Width           =  495
```

```
  End
  Begin VB.Label Label1
    Caption         =   "โปรแกรมนับรอยอนุภาคแอลฟาบนฟิลม์เซลลูโลสไนเตรต"
    BeginProperty Font
      name            =   "CordiaUPC"
      charset         =   1
      weight          =   700
      size            =   18
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    Height          =   375
    Left            =   360
    TabIndex        =   0
    Top             =   480
    Width           =   6255
  End
End
Attribute VB_Name = "Form1"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Private Sub Text1_Change()

End Sub


Private Sub Form_Click()
    Unload Form1
    Load BinContForm
    BinContForm.Show
End Sub
```

ไฟล์ bincont1.frm

```
VERSION 4.00
Begin VB.Form BinContForm
   Caption         =   "โปรแกรมแปลงภาพเป็นขาว-ดำ"
   ClientHeight    =   6630
   ClientLeft      =   45
   ClientTop       =   1800
   ClientWidth     =   9600
   Height          =   7320
   Left            =   -15
   LinkTopic       =   "Form1"
   ScaleHeight     =   442
   ScaleMode       =   3  'Pixel
   ScaleWidth      =   640
   Top             =   1170
   Width           =   9720
   WindowState     =   2  'Maximized
   Begin VB.PictureBox HistPict
      AutoRedraw      =   -1  'True
      Height          =   2775
      Left            =   5400
      MousePointer    =   2  'Cross
      ScaleHeight     =   183
      ScaleMode       =   3  'Pixel
      ScaleWidth      =   260
      TabIndex        =   4
      Top             =   120
      Width           =   3930
   End
   Begin VB.PictureBox FromSwin
      Height          =   6495
      Left            =   0
      ScaleHeight     =   431
      ScaleMode       =   3  'Pixel
      ScaleWidth      =   335
      TabIndex        =   2
      Top             =   0
      Width           =   5055
      Begin VB.PictureBox FromPict
         AutoRedraw      =   -1  'True
         AutoSize        =   -1  'True
         Height          =   1905
         Left            =   0
         ScaleHeight     =   125
         ScaleMode       =   3  'Pixel
         ScaleWidth      =   90
         TabIndex        =   3
         Top             =   0
```

```
    Width          =   1380
  End
End
Begin VB.HScrollBar FromHBar
  Enabled        =   0   'False
  Height         =   255
  Left           =   0
  TabIndex       =   1
  Top            =   3840
  Width          =   3765
End
Begin VB.VScrollBar FromVBar
  Enabled        =   0   'False
  Height         =   3855
  Left           =   3720
  TabIndex       =   0
  Top            =   0
  Width          =   255
End
Begin MSComDlg.CommonDialog FileDialog
  Left           =   3840
  Top            =   0
  _version       =   65536
  _extentx       =   847
  _extenty       =   847
  _stockprops    =   0
  cancelerror    =   -1  'True
End
Begin VB.Menu MnuFile
  Caption        =   "&File"
  Begin VB.Menu mnuFileLoad
    Caption        =   "&Load..."
    Shortcut       =   ^L
  End
  Begin VB.Menu mnuFileSep2
    Caption        =   "-"
  End
  Begin VB.Menu mnuFileSaveAs
    Caption        =   "&Save As"
  End
  Begin VB.Menu MnuFileExit
    Caption        =   "&Exit"
  End
End
Begin VB.Menu MnuGo
  Caption        =   "&Goto"
  Begin VB.Menu MnuFilter
    Caption        =   "&Filter Program"
  End
```

```
    Begin VB.Menu MnuCount
      Caption         =   "&Count Program"
    End
  End
End
Attribute VB_Name = "BinContForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Dim SysPalSize As Integer
Dim NumStaticColors As Integer
Dim StaticColor1 As Integer
Dim StaticColor2 As Integer
Dim DataChanged As Boolean
Dim FileLoaded As String
Dim LogPal As Integer
Dim bytes() As Byte
Dim wid As Long
Dim hgt As Long
 Dim origpal(0 To 255) As PALETTEENTRY
 Dim newpal(0 To 255) As PALETTEENTRY
Dim IndexCounts(0 To 255) As Long
Dim ValueCounts(0 To 255) As Long
 Dim palentry(0 To 255) As PALETTEENTRY
Dim MaxBlack As Integer

' ****************************************************
' Adjust the colors in the image so all those with
' original brightness greater than max_black are
' white and all others are black.
' ****************************************************
Sub SetBinaryContrast(pic As Control, max_black As Integer)
Dim i As Integer
Dim val As Integer
Dim status As Long

    MaxBlack = max_black

    ' Remap the values in the logical palette.
    For i = StaticColor1 + 1 To StaticColor2 - 1
        ' Assumes this is a gray scale image.
        If origpal(i).peRed > max_black Then
            val = 255
        Else
            val = 0
        End If
        With newpal(i)
            .peRed = val
```

```
            .peGreen = val
            .peBlue = val
            .peFlags = PC_NOCOLLAPSE
        End With
    Next i
    i = SetPaletteEntries(LogPal, StaticColor1 + 1, StaticColor2 - StaticColor1 - 1, newpal(StaticColor1 + 1))
    i = RealizePalette(pic.hdc)
End Sub

' ***************************************************
' Return the index of the nonstatic gray closest
' to the given value (assuming the non-static
' colors are a gray scale created by
' MatchGrayPalette).
' ***************************************************
Function NearestNonstaticGray(c As Integer) As Integer
Dim dgray As Single

    If c < 0 Then
        c = 0
    ElseIf c > 255 Then
        c = 255
    End If
    dgray = 255 / (StaticColor2 - StaticColor1 - 2)
    NearestNonstaticGray = c / dgray + StaticColor1 + 1
End Function


' ***************************************************
' Count the number of pixels with each palette
' index.
' ***************************************************
Sub CountIndexes()
Dim X As Integer
Dim Y As Integer
Dim idx As Integer

    ' Start from scratch.
    For X = 0 To SysPalSize - 1
        IndexCounts(X) = 0
    Next X

    ' Count the indexes.
    For Y = 1 To hgt
        For X = 1 To wid
            idx = bytes(X, Y)
            IndexCounts(idx) = IndexCounts(idx) + 1
        Next X
    Next Y
```

```
End Sub

' ************************************************
' Count the brightness values.
' ************************************************
Sub CountValues()
Dim i As Integer
Dim val As Integer
Dim brightval As Integer
Dim darkval As Integer

    ' Start from scratch.
    For i = 0 To SysPalSize - 1
        ValueCounts(i) = 0
    Next i

    ' Add up the values.
    '
    ' For each palette index i, with brightness
    ' val, there are IndexCounts(i) pixels with
    ' that brightness.
    For i = 0 To SysPalSize - 1
        With origpal(i)
            val = (CInt( peRed) + .peGreen + .peBlue) / 3
        End With
        ValueCounts(val) = ValueCounts(val) + IndexCounts(i)
    Next i
End Sub

' ************************************************
' Load the indicated file and prepare to work
' with its palette.
' ************************************************
Sub LoadFromPict(fname As String)
Dim i As Integer

    On Error GoTo LoadFileError
    FromPict.Picture = LoadPicture(fname)
    On Error GoTo 0

    FromHBar.Enabled = False
    FromVBar.Enabled = False
    DoEvents

    MatchGrayPalette FromPict

    FromPict.Move 0, 0
    ResetScrollBars
```

```
    ' Make the new and original palettes match.
    For i = 0 To SysPalSize - 1
        newpal(i) = origpal(i)
    Next i

    ' Count the pixels with each palette index.
    CountIndexes

    ' Count the brightness values.
    CountValues

    ' Display the current histogram.
    MaxBlack = -1
    ShowHistogram

    Caption = "Binary Contrast [" & fname & "]"
    Exit Sub

LoadFileError:
    Beep
    MsgBox "Error loading file " & fname & "." & _
        vbCrLf & Error$
    Exit Sub
End Sub


Private Sub Form_Unload(CANCEL As Integer)
  SavePicture FromPict.Image, ("c:\sketchup\temp.bmp")

End Sub


' ***************************************************
' Load the control's palette so it matches the
' the system palette. Remap any of the image's
' pixels that use static colors to non-static
' colors.
'
' Set the following module global variables.
'   LogPal      Image logical palette handle
'   origpal()   Image logical palette entries.
'   wid         Width of image.
'   hgt         Height of image.
'   bytes(1 To wid, 1 To hgt)
'               Image pixel values.
' ***************************************************
Sub MatchColorPalette(pic As Control)
Dim sys(0 To 255) As PALETTEENTRY
Dim i As Integer
```

```
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim X As Integer
Dim Y As Integer
Dim clr As Integer

    ' Make sure pic has the foreground palette.
    pic.ZOrder
    i = RealizePalette(pic.hdc)
    DoEvents

    ' Get the system palette entries.
    i = GetSystemPaletteEntries(pic.hdc, 0, SysPalSize, sys(0))

    ' Make the logical palette as big as possible.
    LogPal = pic.Picture.hPal
    If ResizePalette(LogPal, SysPalSize) = 0 Then
        Beep
        MsgBox "Error resizing logical palette.", _
            vbExclamation
        Exit Sub
    End If

    ' Blank the non-static colors.
    For i = 0 To StaticColor1
        origpal(i) = sys(i)
    Next i
    For i = StaticColor1 + 1 To StaticColor2 - 1
        With origpal(i)
            .peRed = 0
            .peGreen = 0
            .peBlue = 0
            .peFlags = PC_NOCOLLAPSE
        End With
    Next i
    For i = StaticColor2 To 255
        origpal(i) = sys(i)
    Next i
    i = SetPaletteEntries(LogPal, 0, SysPalSize, origpal(0))

    ' Insert the non-static colors.
    For i = StaticColor1 + 1 To StaticColor2 - 1
        origpal(i) = sys(i)
        origpal(i).peFlags = PC_NOCOLLAPSE
    Next i
    i = SetPaletteEntries(LogPal, StaticColor1 + 1, StaticColor2 - StaticColor1 - 1, origpal(StaticColor1 + 1))

    ' Realize the new palette.
```

```
    i = RealizePalette(pic.hdc)

    ' Get the image pixels.
    hbm = pic.Image
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight
    ReDim bytes(1 To wid, 1 To hgt)
    status = GetBitmapBits(hbm, wid * hgt, bytes(1, 1))

    ' Remap any pixels using static colors.
    For Y = 1 To hgt
        For X = 1 To wid
            clr = bytes(X, Y)
            If clr <= StaticColor1 Or clr >= StaticColor2 Then
                With sys(clr)
                    bytes(X, Y) = _
                        NearestNonstaticColor( _
                        .peRed, .peGreen, .peBlue)
                End With
            End If
        Next X
    Next Y

    ' Update the image's pixel values.
    status = SetBitmapBits(hbm, wid * hgt, bytes(1, 1))

    pic.Refresh
End Sub


' ************************************************
' Load the control's palette so the non-static
' colors are grays. Map the logical palette to
' match the system palette. Convert the image to
' use the non-static grays.
'
' Set the following module global variables.
'   LogPal      Image logical palette handle.
'   origpal()   Image logical palette entries.
'   wid         Width of image.
'   hgt         Height of image.
'   bytes(1 To wid, 1 To hgt)
'               Image pixel values.
' ************************************************
Sub MatchGrayPalette(pic As Control)
Dim sys(0 To 255) As PALETTEENTRY
```

```
Dim i As Integer
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim X As Integer
Dim Y As Integer
Dim gray As Single
Dim dgray As Single
Dim c As Integer
Dim clr As Integer

    ' Make sure pic has the foreground palette.
    pic.ZOrder
    i = RealizePalette(pic.hdc)
    DoEvents

    ' Get the system palette entries.
    i = GetSystemPaletteEntries(pic.hdc, 0, SysPalSize, sys(0))

    ' Get the image pixels.
    hbm = pic.Image
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight
    ReDim bytes(1 To wid, 1 To hgt)
    status = GetBitmapBits(hbm, wid * hgt, bytes(1, 1))

    ' Make the logical palette as big as possible.
    LogPal = pic.Picture.hPal
    If ResizePalette(LogPal, SysPalSize) = 0 Then
        Beep
        MsgBox "Error resizing logical palette.", _
            vbExclamation
        Exit Sub
    End If

    ' Blank the non-static colors.
    For i = 0 To StaticColor1
        origpal(i) = sys(i)
    Next i
    For i = StaticColor1 + 1 To StaticColor2 - 1
        With origpal(i)
            .peRed = 0
            .peGreen = 0
            .peBlue = 0
            .peFlags = PC_NOCOLLAPSE
        End With
    Next i
    For i = StaticColor2 To 255
```

```
    origpal(i) = sys(i)
  Next i
  i = SetPaletteEntries(LogPal, 0, SysPalSize, origpal(0))

  ' Insert the non-static grays.
  gray = 0
  dgray = 255 / (StaticColor2 - StaticColor1 - 2)
  For i = StaticColor1 + 1 To StaticColor2 - 1
    c = gray
    gray = gray + dgray
    With origpal(i)
      .peRed = c
      .peGreen = c
      .peBlue = c
    End With
  Next i
  i = SetPaletteEntries(LogPal, StaticColor1 + 1, StaticColor2 - StaticColor1 - 1, origpal(StaticColor1 + 1))

  ' Recreate the image using the new colors.
  For Y = 1 To hgt
    For X = 1 To wid
      clr = bytes(X, Y)
      With sys(clr)
        c = (CInt(.peRed) + .peGreen + .peBlue) / 3
      End With
      bytes(X, Y) = NearestNonstaticGray(c)
    Next X
  Next Y

  status = SetBitmapBits(hbm, wid * hgt, bytes(1, 1))

  ' Realize the gray palette.
  i = RealizePalette(pic.hdc)

  pic.Refresh
End Sub


' ***************************************************
' Return the index of the nonstatic color closest
' to the given color value.
' ***************************************************
Function NearestNonstaticColor(ByVal r As Integer, ByVal g As Integer, ByVal b As Integer) As Integer
Dim best_i As Integer
Dim best_dist As Long
Dim dist As Long
Dim dr As Long
Dim dg As Long
Dim db As Long
```

```
Dim i As Integer

    best_dist = 1000000
    For i = StaticColor1 + 1 To StaticColor2 - 1
        With origpal(i)
            dr = r - .peRed
            dg = g - .peGreen
            db = b - .peBlue
            dist = dr * dr + dg * dg + db * db
        End With
        If best_dist > dist Then
            best_i = i
            best_dist = dist
        End If
    Next i
    NearestNonstaticColor = best_i
End Function



' ***************************************************
' Create a brightness histogram for the image.
' ***************************************************
Sub ShowHistogram()
Dim i As Integer
Dim maxy As Single
Dim yscale As Single

    HistPict.Cls
    HistPict.Refresh

    ' ***********************
    ' * Display the output *
    ' ***********************
    For i = 0 To SysPalSize - 1
        If maxy < ValueCounts(i) Then _
            maxy = ValueCounts(i)
    Next i
    If maxy <> 0 Then
        yscale = -HistPict.ScaleHeight / maxy
    Else
        yscale = 0
    End If

    maxy = HistPict.ScaleTop + HistPict.ScaleHeight
    For i = 0 To SysPalSize - 1
        HistPict.Line (i + 1, maxy)-Step(0, ValueCounts(i) * yscale)
    Next i
```

```
    ' Draw a line at the cutoff.
    If MaxBlack >= 0 Then _
        HistPict.Line (MaxBlack + 1, 0)- _
            Step(0, HistPict.ScaleHeight), vbRed
End Sub
' ************************************************
' Set the Max and LargeChange properties for the
' image scroll bars.
' ************************************************
Sub ResetScrollBars()
    ' FromHBar.
    FromHBar.value = 0
    If FromSwin.ScaleWidth >= FromPict.Width Then
        FromHBar.Enabled = False
    Else
        FromHBar.Max = FromPict.Width - FromSwin.ScaleWidth
        FromHBar.LargeChange = FromSwin.ScaleWidth
        FromHBar.Enabled = True
    End If

    ' FromVBar.
    FromVBar.value = 0
    If FromSwin.ScaleHeight >= FromPict.Height Then
        FromVBar.Enabled = False
    Else
        FromVBar.Max = FromPict.Height - FromSwin.ScaleHeight
        FromVBar.LargeChange = FromSwin.ScaleHeight
        FromVBar.Enabled = True
    End If
End Sub



' ************************************************
' Give the form and all the picture boxes an
' hourglass cursor.
' ************************************************
Sub WaitStart()
    MousePointer = vbHourglass
    FromPict.MousePointer = vbHourglass
    HistPict.MousePointer = vbHourglass
    DoEvents
End Sub



' ************************************************
' Restore the mouse pointers for the form and all
' the picture boxes.
```

```
' ************************************************
Sub WaitEnd()
    MousePointer = vbDefault
    FromPict.MousePointer = vbDefault
    HistPict.MousePointer = vbDefault
End Sub

Private Sub Form_Load()
    ' Make sure the screen supports palettes.
    If Not GetDeviceCaps(hdc, RASTERCAPS) And RC_PALETTE Then
        Beep
        MsgBox "This monitor does not support palettes.", _
            vbCritical
        End
    End If

    ' Get system palette size and # static colors.
    SysPalSize = GetDeviceCaps(hdc, SIZEPALETTE)
    NumStaticColors = GetDeviceCaps(hdc, NUMRESERVED)
    StaticColor1 = NumStaticColors \ 2 - 1
    StaticColor2 = SysPalSize - NumStaticColors \ 2

    ' Remove the borders from FromPict.
    FromPict.BorderStyle = vbTransparent

    ' Make sure FromPict has control.
    FromPict.ZOrder

End Sub

' ************************************************
' Make the picture as large as possible.
' ************************************************
Private Sub Form_Resize()
Const GAP = 4

Dim hgt As Single
Dim wid As Single

    If WindowState = vbMinimized Then Exit Sub

    hgt = ScaleHeight - FromHBar.Height - 1
    wid = ScaleWidth - FromVBar.Width - 1 - _
        GAP - HistPict.Width

    ' Place FromSwin and its scroll bars.
    FromSwin.Move 0, 0, wid, hgt
    FromVBar.Move _
        FromSwin.Left + FromSwin.Width + 1, _
```

```
        0, FromVBar.Width, hgt
    FromHBar.Move _
        FromSwin.Left, FromSwin.Height + 1, _
        wid

    ' Place HistPict.
    HistPict.Move FromVBar.Left + FromVBar.Width + GAP, 0

    ' Set the scroll bar limits.
    ResetScrollBars
End Sub

' ***************************************************
' Move FromPict within FromSwin.
' ***************************************************
Private Sub FromHBar_Change()
    FromPict.Left = -FromHBar.value
End Sub

' ***************************************************
' Move FromPict within FromSwin.
' ***************************************************
Private Sub FromHBar_Scroll()
    FromPict.Left = -FromHBar.value
End Sub


' ***************************************************
' Set the binary contrast cutoff at this X value.
' ***************************************************
Private Sub HistPict_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If X > 255 Then X = 256
    SetBinaryContrast FromPict, X - 1
    ShowHistogram
End Sub

Private Sub MnuCount_Click()
    Unload BinContForm
    Load Prepare
    Prepare.Show

End Sub

' ***************************************************
' Load a new image file.
' ***************************************************
Private Sub mnuFileLoad_Click()
Dim fname As String
```

```
    ' Allow the user to pick a file.
    On Error Resume Next
    FileDialog.filename = "*.BMP;*.ICO;*.RLE;*.WMF;* DIB"
    FileDialog.Flags = cdlOFNFileMustExist + cdlOFNHideReadOnly
    FileDialog.ShowOpen
    If Err.Number = cdlCancel Then
        Exit Sub
    ElseIf Err.Number <> 0 Then
        Beep
        MsgBox "Error selecting file.", , vbExclamation
        Exit Sub
    End If
    On Error GoTo 0

    fname = Trim$(FileDialog.filename)
    FileDialog.InitDir = Left$(fname, Len(fname) _
        - Len(FileDialog.FileTitle) - 1)

    ' Load the picture.
    WaitStart
    DoEvents
    LoadFromPict fname
    WaitEnd
End Sub



' ****************************************************
' End the application. (See also the QueryUnload
' event.)
' ****************************************************
Private Sub mnuFileExit_Click()
 End

End Sub



' ****************************************************
' Move FromPict within FromSwin.
' ****************************************************
Private Sub FromVBar_Change()
    FromPict.Top = -FromVBar.value
End Sub



' ****************************************************
' Move FromPict within FromSwin.
```

```
' ************************************************
Private Sub FromVBar_Scroll()
    FromPict.Top = -FromVBar.value
End Sub

' ************************************************
' Save the image in a new file.
' ************************************************
Private Sub mnuFileSaveAs_Click()
Dim fname As String

    ' Allow the user to pick a file.
    On Error Resume Next
    FileDialog.filename = "*.BMP;*.ICO;*.RLE;*.WMF;*.DIB"
    FileDialog.Flags = cdlOFNOverwritePrompt + _
        cdlOFNHideReadOnly + cdlOFNPathMustExist
    FileDialog.ShowSave
    If Err.Number = cdlCancel Then
        Exit Sub
    ElseIf Err.Number <> 0 Then
        Beep
        MsgBox "Error selecting file.", , vbExclamation
        Exit Sub
    End If
    On Error GoTo 0

    fname = Trim$(FileDialog.filename)
    FileDialog.InitDir = Left$(fname, Len(fname) _
        - Len(FileDialog.FileTitle) - 1)

    ' Save the picture.
    WaitStart
    DoEvents
    SaveFromPict fname
    WaitEnd
End Sub

' ************************************************
' Save the picture in the indicated file.
' ************************************************
Sub SaveFromPict(fname As String)
    On Error GoTo SaveError
    SavePicture FromPict.Image, fname
```

```
    Caption = "Filters [" & fname & "]"
    FileLoaded = fname
    DataChanged = False
    Exit Sub

SaveError:
    Beep
    MsgBox "Error saving picture in file " & _
        fname & "." & vbCrLf & vbCrLf & _
        Error$, , vbExclamation
    Exit Sub

End Sub




Private Sub MnuFilt_Click()

End Sub

Private Sub mnuFilter_Click()
   Unload BinContForm
    Load FilterForm
    FilterForm.Show
End Sub
```

ไฟล์ filter1.frm

```
VERSION 4.00
Begin VB.Form FilterForm
   Caption        =  "โปรแกรมตกแต่งภาพ"
   ClientHeight   =  4020
   ClientLeft     =  840
   ClientTop      =  1275
   ClientWidth    =  8310
   Height         =  4710
   Left           =  780
   LinkTopic      =  "Form1"
   ScaleHeight    =  268
   ScaleMode      =  3  'Pixel
   ScaleWidth     =  554
   Top            =  645
   Width          =  8430
   WindowState    =  2  'Maximized
   Begin VB.ComboBox FilterCombo
      Height         =  315
      Left           =  3360
      Style          =  2  'Dropdown List
      TabIndex       =  11
      Top            =  480
      Width          =  1575
   End
   Begin VB.PictureBox ToSwin
      Height         =  3735
      Left           =  5040
      ScaleHeight    =  247
      ScaleMode      =  3  'Pixel
      ScaleWidth     =  199
      TabIndex       =  9
      Top            =  0
      Width          =  3015
      Begin VB.PictureBox ToPict
         AutoRedraw     =  -1  'True
         AutoSize       =  -1  'True
         Height         =  45
         Left           =  0
         MousePointer   =  2  'Cross
         Picture        =  "FILTER1.frx":0000
         ScaleHeight    =  1
         ScaleMode      =  3  'Pixel
         ScaleWidth     =  1
         TabIndex       =  10
         Top            =  0
         Width          =  45
      End
```

```
End
Begin VB.PictureBox FromSwin
   Height          =   3735
   Left            =   0
   ScaleHeight     =   247
   ScaleMode       =   3  'Pixel
   ScaleWidth      =   199
   TabIndex        =   7
   Top             =   0
   Width           =   3015
   Begin VB.PictureBox FromPict
      AutoRedraw      =   -1  'True
      AutoSize        =   -1  'True
      Height          =   45
      Left            =   0
      MousePointer    =   2  'Cross
      Picture         =   "FILTER1.frx":0446
      ScaleHeight     =   1
      ScaleMode       =   3  'Pixel
      ScaleWidth      =   1
      TabIndex        =   8
      Top             =   0
      Width           =   45
   End
End
Begin VB.VScrollBar ToVBar
   Height          =   3735
   Left            =   8040
   TabIndex        =   6
   Top             =   0
   Width           =   255
End
Begin VB.HScrollBar ToHBar
   Height          =   255
   Left            =   5040
   TabIndex        =   5
   Top             =   3720
   Width           =   3045
End
Begin VB.CommandButton CmdCopy
   Caption         =   "<-- Copy"
   Enabled         =   0  'False
   Height          =   495
   Left            =   3720
   TabIndex        =   4
   Top             =   1920
   Width           =   855
End
Begin VB.CommandButton CmdApply
```

```
  Caption         =  "Apply ->"
  Enabled         =  0  'False
  Height          =  495
  Left            =  3720
  TabIndex        =  3
  Top             =  1080
  Width           =  855
End
Begin VB.CheckBox ProgressCheck
  Caption         =  "Show Progress"
  Height          =  255
  Left            =  3360
  TabIndex        =  2
  Top             =  120
  Value           =  1  'Checked
  Width           =  1575
End
Begin VB.HScrollBar FromHBar
  Height          =  255
  Left            =  0
  TabIndex        =  1
  Top             =  3720
  Width           =  3045
End
Begin VB.VScrollBar FromVBar
  Height          =  3735
  Left            =  3000
  TabIndex        =  0
  Top             =  0
  Width           =  255
End
Begin MSComDlg.CommonDialog FileDialog
  Left            =  3960
  Top             =  2880
  _version        =  65536
  _extentx        =  847
  _extenty        =  847
  _stockprops     =  0
  cancelerror     =  -1  'True
End
Begin VB.Menu mnuFile
  Caption         =  "&File"
  Begin VB.Menu mnuFileLoad
    Caption       =  "&Load..."
    Shortcut      =  ^L
  End
  Begin VB.Menu mnuFileSave
    Caption       =  "&Save"
    Enabled       =  0  'False
```

```
      Shortcut        =  ^S
    End
    Begin VB.Menu mnuFileSaveAs
      Caption         =  "Save &As..."
      Enabled         =  0   'False
      Shortcut        =  ^A
    End
    Begin VB.Menu mnuFileSep1
      Caption         =  "-"
    End
    Begin VB.Menu mnuFileRevert
      Caption         =  "&Revert"
      Enabled         =  0   'False
      Shortcut        =  ^R
    End
    Begin VB.Menu mnuFileSep2
      Caption         =  "-"
    End
    Begin VB.Menu mnuFileExit
      Caption         =  "E&xit"
    End
  End
  Begin VB.Menu MnuGoto
    Caption         =  "&Goto"
    Begin VB.Menu MnuBicont
      Caption         =  "&Binary Contrast Program"
    End
    Begin VB.Menu MnuCount
      Caption         =  "&Count Program"
    End
  End
End
Attribute VB_Name = "FilterForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Dim SysPalSize As Integer
Dim NumStaticColors As Integer
Dim StaticColor1 As Integer
Dim StaticColor2 As Integer

Dim DataChanged As Boolean
Dim FileLoaded As String

Dim LogPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim wid As Long
Dim hgt As Long
```

```
Dim bytes() As Byte

' ************************************************
' Put the names of the available filters in the
' filter combo box.
' ************************************************
Sub LoadFilterChoices()

    FilterCombo.AddItem "Dilation"
    FilterCombo.AddItem "Dilation Outline"

    FilterCombo.AddItem "Average 3x3"
    FilterCombo.AddItem "Low Pass 3x3"
    FilterCombo.AddItem "Low Pass 5x5"
    FilterCombo.AddItem "Low Pass 7x7"
    FilterCombo.AddItem "High Pass 1"
    FilterCombo.AddItem "High Pass 2"
    FilterCombo.AddItem "High Pass 3"
    FilterCombo.AddItem "High Pass 4"
    FilterCombo.AddItem "Prewitt Up"
    FilterCombo.AddItem "Prewitt Up-Right"
    FilterCombo.AddItem "Prewitt Right"
    FilterCombo.AddItem "Prewitt Down-Right"
    FilterCombo.AddItem "Prewitt Down"
    FilterCombo.AddItem "Prewitt Down-Left"
    FilterCombo.AddItem "Prewitt Left"
    FilterCombo.AddItem "Prewitt Up-Left"
    FilterCombo.AddItem "Laplacian 1"
    FilterCombo.AddItem "Laplacian 2"
    FilterCombo.AddItem "Minimum 3x3"
    FilterCombo.AddItem "Median 3x3"
    FilterCombo.AddItem "Maximum 3x3"
    FilterCombo.AddItem "Voting 3x3"
    FilterCombo.AddItem "Emboss Up"
    FilterCombo.AddItem "Emboss Up-Right"
    FilterCombo.AddItem "Emboss Right"
    FilterCombo.AddItem "Emboss Down-Right"
    FilterCombo.AddItem "Emboss Down"
    FilterCombo.AddItem "Emboss Down-Left"
   FilterCombo.AddItem "Emboss Left"
   FilterCombo.AddItem "Emboss Up-Left"

   FilterCombo.AddItem "Erosion"
   FilterCombo.AddItem "Erosion Outline"

   FilterCombo.ListIndex = 0
End Sub

' ************************************************
```

```
' Load the control's palette so it matches the
' the system palette. Remap any of the image's
' pixels that use static colors to non-static
' colors.
'
' Set the following module global variables.
'   LogPal      Image logical palette handle.
'   palentry()  Image logical palette entries.
'   wid         Width of image.
'   hgt         Height of image.
'   bytes(1 To wid, 1 To hgt)
'               Image pixel values.
' ************************************************
Sub MatchColorPalette(pic As Control)
Dim sys(0 To 255) As PALETTEENTRY
Dim i As Integer
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim X As Integer
Dim Y As Integer
Dim clr As Integer

    ' Make sure pic has the foreground palette.
    pic.ZOrder
    i = RealizePalette(pic.hdc)
    DoEvents

    ' Get the system palette entries.
    i = GetSystemPaletteEntries(pic.hdc, 0, SysPalSize, sys(0))

    ' Make the logical palette as big as possible.
    LogPal = pic.Picture.hPal
    If ResizePalette(LogPal, SysPalSize) = 0 Then
        Beep
        MsgBox "Error resizing logical palette.", _
            vbExclamation
        Exit Sub
    End If

    ' Blank the non-static colors.
    For i = 0 To StaticColor1
        palentry(i) = sys(i)
    Next i
    For i = StaticColor1 + 1 To StaticColor2 - 1
        With palentry(i)
            .peRed = 0
            .peGreen = 0
            .peBlue = 0
```

```
            .peFlags = PC_NOCOLLAPSE
        End With
    Next i
    For i = StaticColor2 To 255
        palentry(i) = sys(i)
    Next i
    i = SetPaletteEntries(LogPal, 0, SysPalSize, palentry(0))

    ' Insert the non-static colors.
    For i = StaticColor1 + 1 To StaticColor2 - 1
        palentry(i) = sys(i)
        palentry(i).peFlags = PC_NOCOLLAPSE
    Next i
    i = SetPaletteEntries(LogPal, StaticColor1 + 1, StaticColor2 - StaticColor1 - 1, palentry(StaticColor1 + 1))

    ' Realize the new palette.
    i = RealizePalette(pic.hdc)

    ' Get the image pixels.
    hbm = pic.Image
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight
    ReDim bytes(1 To wid, 1 To hgt)
    status = GetBitmapBits(hbm, wid * hgt, bytes(1, 1))

    ' Remap any pixels using static colors.
    For Y = 1 To hgt
        For X = 1 To wid
            clr = bytes(X, Y)
            If clr <= StaticColor1 Or clr >= StaticColor2 Then
                With sys(clr)
                    bytes(X, Y) = _
                        NearestNonstaticColor( _
                        .peRed, .peGreen, .peBlue)
                End With
            End If
        Next X
    Next Y

    ' Update the image's pixel values.
    status = SetBitmapBits(hbm, wid * hgt, bytes(1, 1))

    pic.Refresh
End Sub
```

```
' ************************************************
' Load the control's palette so the non-static
' colors are grays. Map the logical palette to
' match the system palette. Convert the image to
' use the non-static grays.
'
' Set the following module global variables.
'   LogPal      Image logical palette handle.
'   palentry()  Image logical palette entries.
'   wid         Width of image.
'   hgt         Height of image.
'   bytes(1 To wid, 1 To hgt)
'               Image pixel values.
' ************************************************
Sub MatchGrayPalette(pic As Control)
Dim sys(0 To 255) As PALETTEENTRY
Dim i As Integer
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim X As Integer
Dim Y As Integer
Dim gray As Single
Dim dgray As Single
Dim c As Integer
Dim clr As Integer

    ' Make sure pic has the foreground palette.
    pic.ZOrder
    i = RealizePalette(pic.hdc)
    DoEvents

    ' Get the system palette entries.
    i = GetSystemPaletteEntries(pic.hdc, 0, SysPalSize, sys(0))

    ' Get the image pixels.
    hbm = pic.Image
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight
    ReDim bytes(1 To wid, 1 To hgt)
    status = GetBitmapBits(hbm, wid * hgt, bytes(1, 1))

    ' Make the logical palette as big as possible.
    LogPal = pic.Picture.hPal
    If ResizePalette(LogPal, SysPalSize) = 0 Then
        Beep
```

```
    MsgBox "Error resizing logical palette.", _
        vbExclamation
    Exit Sub
End If

' Blank the non-static colors.
For i = 0 To StaticColor1
    palentry(i) = sys(i)
Next i
For i = StaticColor1 + 1 To StaticColor2 - 1
    With palentry(i)
        .peRed = 0
        .peGreen = 0
        .peBlue = 0
        .peFlags = PC_NOCOLLAPSE
    End With
Next i
For i = StaticColor2 To 255
    palentry(i) = sys(i)
Next i
i = SetPaletteEntries(LogPal, 0, SysPalSize, palentry(0))

' Insert the non-static grays.
gray = 0
dgray = 255 / (StaticColor2 - StaticColor1 - 2)
For i = StaticColor1 + 1 To StaticColor2 - 1
    c = gray
    gray = gray + dgray
    With palentry(i)
        .peRed = c
        .peGreen = c
        .peBlue = c
    End With
Next i
i = SetPaletteEntries(LogPal, StaticColor1 + 1, StaticColor2 - StaticColor1 - 1, palentry(StaticColor1 + 1))

' Recreate the image using the new colors.
For Y = 1 To hgt
    For X = 1 To wid
        clr = bytes(X, Y)
        With sys(clr)
            c = (CInt(.peRed) + .peGreen + .peBlue) / 3
        End With
        bytes(X, Y) = NearestNonstaticGray(c)
    Next X
Next Y

status = SetBitmapBits(hbm, wid * hgt, bytes(1, 1))
```

```
    ' Realize the gray palette.
    i = RealizePalette(pic.hdc)

    pic.Refresh
End Sub


' ****************************************************
' Return the index of the nonstatic gray closest
' to the given value (assuming the non-static
' colors are a gray scale created by
' MatchGrayPalette).
' ****************************************************
Function NearestNonstaticGray(c As Integer) As Integer
Dim dgray As Single

    If c < 0 Then
        c = 0
    ElseIf c > 255 Then
        c = 255
    End If
    dgray = 255 / (StaticColor2 - StaticColor1 - 2)
    NearestNonstaticGray = c / dgray + StaticColor1 + 1
End Function


' ****************************************************
' Return the index of the nonstatic color closest
' to the given color value.
' ****************************************************
Function NearestNonstaticColor(ByVal r As Integer, ByVal g As Integer, ByVal b As Integer) As Integer
Dim best_i As Integer
Dim best_dist As Long
Dim dist As Long
Dim dr As Long
Dim dg As Long
Dim db As Long
Dim i As Integer

    best_dist = 1000000
    For i = StaticColor1 + 1 To StaticColor2 - 1
        With palentry(i)
            dr = r - .peRed
            dg = g - .peGreen
            db = b - .peBlue
            dist = dr * dr + dg * dg + db * db
        End With
```

```
        If best_dist > dist Then
            best_i = i
            best_dist = dist
        End If
    Next i
    NearestNonstaticColor = best_i
End Function

' ****************************************************
' If the data has been modified, allow the user
' to save the changes or cancel the operation.
' Return True if:
'
'   - The image data has not been changed since
'     it was loaded.
'   - The user saves the changes.
'   - The user says not to save.
'
' Return False otherwise.
' ****************************************************
Function DataSafe() As Boolean
    DataSafe = True

    ' This is done in a while loop in case the
    ' user starts a save and then cancels.
    Do While DataChanged
        Select Case MsgBox("The data has been modified. Do you want to save the changes?", vbQuestion + vbYesNoCancel, "Data Modified")
            Case vbYes
                If FileLoaded <> "" Then
                    mnuFileSave_Click
                Else
                    mnuFileSaveAs_Click
                End If
                DataSafe = Not DataChanged

            Case vbNo
                DataSafe = True
                Exit Do

            Case vbCancel
                DataSafe = False
                Exit Do
        End Select
```

```
    Loop
End Function


' ****************************************************
' Load the indicated file and prepare to work
' with its palette.
' ****************************************************
Sub LoadFromPict(fname As String)
    On Error GoTo LoadFileError
    FromPict.Picture = LoadPicture(fname)

    FromPict.Move 0, 0
    ToPict.Move 0, 0

    MatchGrayPalette FromPict
    ToPict.Picture = FromPict.Image
    MatchGrayPalette ToPict

    FromSwin.ZOrder
    DoEvents
    ToSwin.ZOrder
    DoEvents

    ResetScrollBars
    FileLoaded = fname
    Caption = "Filters [" & fname & "]"
    mnuFileSave.Enabled = True
    mnuFileSaveAs.Enabled = True
    mnuFileRevert.Enabled = True
    CmdApply.Enabled = True
    CmdCopy.Enabled = True
    DataChanged = False
    Exit Sub

LoadFileError:
    Beep
    MsgBox "Error loading file " & fname & "." & _
        vbCrLf & Error$
    Exit Sub
End Sub

' ****************************************************
' Set the Max and LargeChange properties for the
' image scroll bars.
' ****************************************************
Sub ResetScrollBars()
    ' FromHBar
    FromHBar.value = 0
```

```
    If FromSwin.ScaleWidth >= FromPict.Width Then
        FromHBar.Enabled = False
    Else
        FromHBar.Max = FromPict.Width - FromSwin.ScaleWidth
        FromHBar.LargeChange = FromSwin.ScaleWidth
        FromHBar.Enabled = True
    End If

    ' FromVBar.
    FromVBar.value = 0
    If FromSwin.ScaleHeight >= FromPict.Height Then
        FromVBar.Enabled = False
    Else
        FromVBar.Max = FromPict.Height - FromSwin.ScaleHeight
        FromVBar.LargeChange = FromSwin.ScaleHeight
        FromVBar.Enabled = True
    End If

    ' ToHBar.
    ToHBar.value = 0
    If FromSwin.ScaleWidth >= ToPict.Width Then
        ToHBar.Enabled = False
    Else
        ToHBar.Max = ToPict.Width - ToSwin.ScaleWidth
        ToHBar.LargeChange = ToSwin.ScaleWidth
        ToHBar.Enabled = True
    End If

    ' ToVBar.
    ToVBar.value = 0
    If FromSwin.ScaleHeight >= ToPict.Height Then
        ToVBar.Enabled = False
    Else
        ToVBar.Max = ToPict.Height - ToSwin.ScaleHeight
        ToVBar.LargeChange = ToSwin.ScaleHeight
        ToVBar.Enabled = True
    End If
End Sub


' ************************************************
' Give the form and all the picture boxes an
' hourglass cursor
' ************************************************
Sub WaitStart()
    MousePointer = vbHourglass
    FromPict.MousePointer = vbHourglass
    ToPict.MousePointer = vbHourglass
    DoEvents
```

```
End Sub

' ************************************************
' Restore the mouse pointers for the form and all
' the picture boxes.
' ************************************************
Sub WaitEnd()
    MousePointer = vbDefault
    FromPict.MousePointer = vbDefault
    ToPict.MousePointer = vbDefault
End Sub

' ************************************************
' Apply the selected filter to FromPict.
' ************************************************
Private Sub CmdApply_Click()
Static btn_caption As String

Dim fil As New Filter

    ' If the filter is running, stop it.
    If OperationRunning Then
        ' Set a flag so the filter will stop.
        OperationRunning = False

        ' Disable this button.
        CmdApply.Enabled = False
        CmdApply.Caption = "Stopping"
        Exit Sub
    End If

    ' Make sure something is selected.
    If FilterCombo.ListIndex < 0 Then
        Beep
        Exit Sub
    End If

    ' Otherwise start the filter running.
    OperationRunning = True
    btn_caption = CmdApply.Caption
    CmdApply.Caption = "Stop"
    CmdCopy.Enabled = False

    WaitStart

    ' Initialize the filter.
    fil.InitializeFilter FilterCombo.List(FilterCombo.ListIndex)
```

```
    ' Apply the filter.
    fil.ApplyFilter FromPict, ToPict, _
        (ProgressCheck.value = vbChecked)

    ' Reenable this button.
    CmdApply.Caption = btn_caption
    CmdApply.Enabled = True
    CmdCopy.Enabled = True
    OperationRunning = False

    WaitEnd

    ' This could have taken a long time so wake
    ' the user up.
    Beep
End Sub

' ************************************************
' Copy ToPict into FromPict.
' ************************************************
Private Sub CmdCopy_Click()

    FromPict.PaintPicture ToPict.Image, 0, 0
    DataChanged = True
End Sub



' ************************************************
' 1. Make sure we can handle palettes.
' 2. Find out how big the system palette is and how
' many static colors there are.
' 3. Load and display the system palette.
' ************************************************
Private Sub Form_Load()
    ' Make sure the screen supports palettes.
    If Not GetDeviceCaps(hdc, RASTERCAPS) And RC_PALETTE Then
        Beep
        MsgBox "This monitor does not support palettes.", _
            vbCritical
        End
    End If

    ' Get system palette size and # static colors.
    SysPalSize = GetDeviceCaps(hdc, SIZEPALETTE)
    NumStaticColors = GetDeviceCaps(hdc, NUMRESERVED)
    StaticColor1 = NumStaticColors \ 2 - 1
    StaticColor2 = SysPalSize - NumStaticColors \ 2
```

```
    ' Remove the borders from the drawing areas.
    FromPict.BorderStyle = vbTransparent
    ToPict.BorderStyle = vbTransparent

    ' Load the filter choices.
    LoadFilterChoices


End Sub
' ************************************************
' Refuse to unload if there are unsaved changes.
' ************************************************
Private Sub Form_QueryUnload(CANCEL As Integer, UnloadMode As Integer)
    CANCEL = Not DataSafe()
End Sub


' ************************************************
' Make the picture as large as possible.
' ************************************************
Private Sub Form_Resize()
Const GAP = 4

Dim hgt As Single
Dim wid As Single

    If WindowState = vbMinimized Then Exit Sub

    hgt = ScaleHeight - FromHBar.Height - 1
    wid = (ScaleWidth - ProgressCheck.Width - 1 - _
        2 * GAP - 2 * FromVBar.Width - 2) / 2

    ' Place FromSwin and its scroll bars.
    FromSwin.Move 0, 0, wid, hgt
    FromVBar.Move _
        FromSwin.Left + FromSwin.Width + 1, _
        0, FromVBar.Width, hgt
    FromHBar.Move _
        FromSwin.Left, FromSwin.Height + 1, _
        wid

    ' Place the command buttons and stuff.
    ProgressCheck.Left = (ScaleWidth - ProgressCheck.Width) / 2
    FilterCombo.Left = (ScaleWidth - FilterCombo.Width) / 2
    CmdApply.Left = (ScaleWidth - CmdApply.Width) / 2
    CmdCopy.Left = (ScaleWidth - CmdCopy.Width) / 2

    ' Place ToSwin and its scroll bars.
```

```
    ToSwin.Move ProgressCheck.Left + _
        ProgressCheck.Width + GAP, 0, wid, hgt
    ToVBar.Move _
        ToSwin.Left + ToSwin.Width + 1, _
        0, ToVBar.Width, hgt
    ToHBar.Move _
        ToSwin.Left, ToSwin.Height + 1, _
        wid

    ' Set the scroll bar limits.
    ResetScrollBars
End Sub

' ***************************************************
' Move FromPict within FromSwin.
' ***************************************************
Private Sub FromHBar_Change()
    FromPict.Left = -FromHBar.value
End Sub

' ***************************************************
' Move FromPict within FromSwin.
' ***************************************************
Private Sub FromHBar_Scroll()
    FromPict.Left = -FromHBar.value
End Sub

Private Sub MnuBicont_Click()
    Unload FilterForm
    Load BinContForm
    BinContForm.Show
End Sub

Private Sub MnuCount_Click()
    Unload FilterForm
    Load Prepare
    Prepare.Show
End Sub

' ***************************************************
' Load a new image file.
' ***************************************************
Private Sub mnuFileLoad_Click()
Dim fname As String

    ' Make sure any changes have been saved.
    If Not DataSafe() Then Exit Sub

    ' Allow the user to pick a file.
```

```
    On Error Resume Next
    FileDialog.filename = "*.BMP;*.ICO;*.RLE;*.WMF;*.DIB"
    FileDialog.Flags = cdlOFNFileMustExist + cdlOFNHideReadOnly
    FileDialog.ShowOpen
    If Err.Number = cdlCancel Then
        Exit Sub
    ElseIf Err.Number <> 0 Then
        Beep
        MsgBox "Error selecting file.", , vbExclamation
        Exit Sub
    End If
    On Error GoTo 0

    fname = Trim$(FileDialog.filename)
    FileDialog.InitDir = Left$(fname, Len(fname) _
        - Len(FileDialog.FileTitle) - 1)

    ' Load the picture.
    WaitStart
    LoadFromPict fname
    WaitEnd
End Sub

' ***************************************************
' Reload the file.
' ***************************************************
Private Sub mnuFileRevert_Click()
    ' If the data has changed, get confirmation.
    If DataChanged Then
        If MsgBox("The data has been modified. Are you sure you want to remove the changes?", _
            vbQuestion + vbYesNo) = vbNo Then _
                Exit Sub
    End If

    ' Reload the picture.
    WaitStart
    DoEvents
    LoadFromPict FileLoaded
    WaitEnd
End Sub

' ***************************************************
' Save the image in the file from which it was
' loaded.
' ***************************************************
Private Sub mnuFileSave_Click()
    WaitStart
    DoEvents
    SaveFromPict FileLoaded
```

```
    WaitEnd
End Sub

' ************************************************
' Save the image in a new file.
' ************************************************
Private Sub mnuFileSaveAs_Click()
Dim fname As String

    ' Allow the user to pick a file.
    On Error Resume Next
    FileDialog.filename = "*.BMP;*.ICO;*.RLE;*.WMF;*.DIB"
    FileDialog.Flags = cdlOFNOverwritePrompt + _
        cdlOFNHideReadOnly + cdlOFNPathMustExist
    FileDialog.ShowSave
    If Err.Number = cdlCancel Then
        Exit Sub
    ElseIf Err.Number <> 0 Then
        Beep
        MsgBox "Error selecting file.", , vbExclamation
        Exit Sub
    End If
    On Error GoTo 0

    fname = Trim$(FileDialog.filename)
    FileDialog.InitDir = Left$(fname, Len(fname) _
        - Len(FileDialog.FileTitle) - 1)

    ' Save the picture.
    WaitStart
    DoEvents
    SaveFromPict fname
    WaitEnd
End Sub


' ************************************************
' Save the picture in the indicated file.
' ************************************************
Sub SaveFromPict(fname As String)
    On Error GoTo SaveError
    SavePicture ToPict.Image, fname

    Caption = "Filters [" & fname & "]"
    FileLoaded = fname
    DataChanged = False
    Exit Sub

SaveError:
```

```
    Beep
    MsgBox "Error saving picture in file " & _
        fname & "." & vbCrLf & vbCrLf & _
        Error$, , vbExclamation
    Exit Sub

End Sub



' ************************************************
' End the application. (See also the QueryUnload
' event.)
' ************************************************

Private Sub mnuFileExit_Click()
    Unload Me
End Sub



' ************************************************
' Move FromPict within FromSwin.
' ************************************************

Private Sub FromVBar_Change()
    FromPict.Top = -FromVBar.value
End Sub


' ************************************************
' Move FromPict within FromSwin.
' ************************************************

Private Sub FromVBar_Scroll()
    FromPict.Top = -FromVBar.value
End Sub



' ************************************************
' Display a message box showing the filter's
' components and weight.
' ************************************************

Private Sub mnuOptShowFilter_Click()
Dim fil As New Filter
```

```
    ' Make sure something is selected.
    If FilterCombo.ListIndex < 0 Then
        Beep
        Exit Sub
    End If

    ' Initialize the filter.
    fil.InitializeFilter FilterCombo.List(FilterCombo.ListIndex)

    ' Show the filter.
    fil.ShowFilter
End Sub

' ************************************************
' Move ToPict within ToSwin.
' ************************************************

Private Sub ToHBar_Change()
    ToPict.Left = -ToHBar.value
End Sub


' ************************************************
' Move ToPict within ToSwin.
' ************************************************

Private Sub ToHBar_Scroll()
    ToPict.Left = -ToHBar.value
End Sub


' ************************************************
' Move ToPict within ToSwin.
' ************************************************

Private Sub ToVBar_Change()
    ToPict.Top = -ToVBar.value
End Sub
```

```
' ************************************************
' Move ToPict within ToSwin.
' ************************************************

Private Sub ToVBar_Scroll()
    ToPict.Top = -ToVBar.value
End Sub
```

ไฟล์ filter1.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Filter"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Bound As Integer
Private Kernel() As Single
Private Wgt As Single
Private Kind As Integer

' Filter types.
Const FILTER_NONE = 0
Const FILTER_LOWPASS = 1
Const FILTER_HIGHPASS = 2
Const FILTER_PREWITT = 3
Const FILTER_LAPLACIAN = 4
Const FILTER_RANK_MIN = 5
Const FILTER_RANK_MEDIAN = 6
Const FILTER_RANK_MAX = 7
Const FILTER_VOTING = 8
Const FILTER_EMBOSS = 9
Const FILTER_MORPHO = 10
Const FILTER_ERODE_OUTLINE = 11
Const FILTER_DILATE_OUTLINE = 12

' Prewitt and embossing filter types.
Const FILTER_UP = 0
Const FILTER_UP_RIGHT = 1
Const FILTER_RIGHT = 2
Const FILTER_DOWN_RIGHT = 3
Const FILTER_DOWN = 4
Const FILTER_DOWN_LEFT = 5
Const FILTER_LEFT = 6
Const FILTER_UP_LEFT = 7

' Morphological filter types.
Const FILTER_EROSION = 10
Const FILTER_DILATION = 11

' ****************************************************
' Apply a voting filter.
' ****************************************************
Private Sub ApplyVoting(from_pict As Object, to_pict As Object, show_progress As Boolean)
```

```
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim bytesin() As Byte
Dim bytesout() As Byte
Dim wid As Long
Dim hgt As Long
Dim i As Integer
Dim j As Integer
Dim k As Integer

Dim hPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim x As Integer
Dim y As Integer
Dim arr_size As Integer
Dim brightness() As Integer
Dim index() As Integer
Dim count() As Integer
Dim value As Integer
Dim idx As Integer
Dim best_i As Integer

    ' ******************************
    ' * Get the input bitmap data *
    ' ******************************
    ' Get a handle to the input bitmap.
    hbm = from_pict.Image

    ' See how big it is.
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight

    ' Get the bits.
    ReDim bytesin(0 To wid - 1, 0 To hgt - 1)
    status = GetBitmapBits(hbm, wid * hgt, bytesin(0, 0))
    ReDim bytesout(0 To wid - 1, 0 To hgt - 1)

    ' ********************
    ' * Apply the filter *
    ' ********************
    ' Get the current color values.
    hPal = from_pict.Picture.hPal
    i = GetPaletteEntries(hPal, 0, 255, palentry(0))

    ' Compute the new color values.
    arr_size = 2 * Bound + 1
    arr_size = arr_size * arr_size
```

```
ReDim brightness(1 To arr_size)
ReDim index(1 To arr_size)
ReDim count(1 To arr_size)

For x = Bound To wid - 1 - Bound
    ' If the operation has been canceled, stop.
    DoEvents
    If Not OperationRunning Then Exit For

    ' If we should show progress, do so.
    If show_progress Then
        status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
        to_pict.Refresh
    End If

    For y = Bound To hgt - 1 - Bound
        ' Load values into the brightness and
        ' index arrays.
        idx = 0
        For i = -Bound To Bound
            For j = -Bound To Bound
                With palentry(bytesin(x + i, y + j))
                    value = CInt(.peRed) + .peGreen + .peBlue
                End With
                For k = 1 To idx
                    If brightness(k) = value Then
                        count(k) = count(k) + 1
                        Exit For
                    End If
                Next k
                If k > idx Then
                    idx = idx + 1
                    count(idx) = 1
                    brightness(idx) = value
                    index(idx) = bytesin(x + i, y + j)
                End If
            Next j
        Next i
        ' See which value got the most votes.
        value = count(1)
        best_i = 1
        For i = 2 To idx
            If value < count(i) Then
                value = count(i)
                best_i = i
            End If
        Next i

        ' Set the new pixel value.
```

```
            bytesout(x, y) = index(best_i)
        Next y
    Next x

    ' ************************
    ' * Display the output *
    ' ************************
    status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
    to_pict.Refresh
End Sub

' ****************************************************
' Apply a median filter.
' ****************************************************
Private Sub ApplyRank(from_pict As Object, to_pict As Object, show_progress As Boolean)
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim bytesin() As Byte
Dim bytesout() As Byte
Dim wid As Long
Dim hgt As Long
Dim i As Integer
Dim j As Integer

Dim hPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim x As Integer
Dim y As Integer
Dim arr_size As Integer
Dim brightness() As Integer
Dim index() As Integer
Dim idx As Integer
Dim tmp As Integer
Dim best_brightness As Long
Dim best_j As Integer

    ' ******************************
    ' * Get the input bitmap data *
    ' ******************************
    ' Get a handle to the input bitmap.
    hbm = from_pict.Image

    ' See how big it is.
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight

    ' Get the bits.
```

```
ReDim bytesin(0 To wid - 1, 0 To hgt - 1)
status = GetBitmapBits(hbm, wid * hgt, bytesin(0, 0))
ReDim bytesout(0 To wid - 1, 0 To hgt - 1)

' ********************
' * Apply the filter *
' ********************
' Get the current color values.
hPal = from_pict.Picture.hPal
i = GetPaletteEntries(hPal, 0, 255, palentry(0))

' Compute the new color values.
arr_size = 2 * Bound + 1
arr_size = arr_size * arr_size
ReDim brightness(1 To arr_size)
ReDim index(1 To arr_size)

For x = Bound To wid - 1 - Bound
    ' If the operation has been canceled, stop.
    DoEvents
    If Not OperationRunning Then Exit For

    ' If we should show progress, do so.
    If show_progress Then
        status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
        to_pict.Refresh
    End If

    For y = Bound To hgt - 1 - Bound
        ' Load values into the brightness and
        ' index arrays.
        idx = 1
        For i = -Bound To Bound
            For j = -Bound To Bound
                With palentry(bytesin(x + i, y + j))
                    brightness(idx) = CInt(.peRed) + .peGreen + .peBlue
                End With
                index(idx) = bytesin(x + i, y + j)
                idx = idx + 1
            Next j
        Next i
        ' Sort the pixels by brightness.
        For i = 1 To arr_size - 1
            best_brightness = brightness(i)
            best_j = i
            For j = i + 1 To arr_size
                If brightness(j) > brightness(i) Then
                    best_brightness = brightness(j)
                    best_j = j
```

```
                End If
            Next j
            ' Swap the i and best_j positions.
            brightness(best_j) = brightness(i)
            brightness(i) = best_brightness
            tmp = index(best_j)
            index(best_j) = index(i)
            index(i) = tmp
        Next i

        ' Set output value based on rank.
        Select Case Kind
            Case FILTER_RANK_MIN
                bytesout(x, y) = index(1)
            Case FILTER_RANK_MEDIAN
                bytesout(x, y) = index(5)
            Case FILTER_RANK_MAX
                bytesout(x, y) = index(9)
        End Select
    Next y
Next x
' ************************
' * Display the output *
' ************************
status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
to_pict.Refresh
End Sub
' ***************************************************
' Create an outline by subtracting an eroded image
' from the original.
' ***************************************************
Private Sub ApplyErodeOutline(from_pict As Object, to_pict As Object, show_progress As Boolean)
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim bytesin() As Byte
Dim bytesout() As Byte
Dim wid As Long
Dim hgt As Long
Dim i As Integer
Dim j As Integer
Dim c As Integer
Dim hPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim x As Integer
Dim y As Integer
    ' ******************************
    ' * Get the input bitmap data *
    ' ******************************
```

```
' Get a handle to the input bitmap.
hbm = from_pict.Image

' See how big it is.
status = GetObject(hbm, BITMAP_SIZE, bm)
wid = bm.bmWidthBytes
hgt = bm.bmHeight

' Get the bits.
ReDim bytesin(0 To wid - 1, 0 To hgt - 1)
status = GetBitmapBits(hbm, wid * hgt, bytesin(0, 0))
ReDim bytesout(0 To wid - 1, 0 To hgt - 1)
' ********************
' * Apply the filter *
' ********************
' Get the current color values.
hPal = from_pict.Picture.hPal
i = GetPaletteEntries(hPal, 0, 255, palentry(0))

' Do the erosion.
For x = Bound To wid - 1 - Bound
    ' If the operation has been canceled, stop.
    DoEvents
    If Not OperationRunning Then Exit For

    ' If we should show progress, do so.
    If show_progress Then
        status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
        to_pict.Refresh
    End If

    For y = Bound To hgt - 1 - Bound
        For i = -Bound To Bound
            For j = -Bound To Bound
                If palentry(bytesin(x + i, y + j)).peRed <> Kernel(i, j) Then Exit For
            Next j
            If j <= Bound Then Exit For
        Next i
        If j <= Bound Then
            c = bytesin(x, y) - (255 - Wgt)
        Else
            c = bytesin(x, y) - Wgt
        End If
        If c < 0 Then c = 0
        bytesout(x, y) = GetNearestPaletteIndex( _
            hPal, RGB(c, c, c) + &H2000000)
    Next y
Next x
```

```
    ' **********************
    ' * Display the output *
    ' **********************
    status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
    to_pict.Refresh
End Sub
' ************************************************
' Display a message box describing dilated
' outlines.
' ************************************************
Sub ShowDilateOutline()
    MsgBox "This filter creates an outline by subtracting" & _
        "the original image from a dilated version.", _
        vbInformation, "Filter Values"
End Sub
' ************************************************
' Display a message box describing edoded
' outlines.
' ************************************************
Sub ShowErodeOutline()
    MsgBox "This filter creates an outline by " & _
        "subtracting an eroded image from the original.", _
        vbInformation, "Filter Values"
End Sub
' ************************************************
' Create an outline by subtracting the original
' image from a dilated image.
' ************************************************
Private Sub ApplyDilateOutline(from_pict As Object, to_pict As Object, show_progress As Boolean)
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim bytesin() As Byte
Dim bytesout() As Byte
Dim wid As Long
Dim hgt As Long
Dim i As Integer
Dim j As Integer
Dim c As Integer
Dim hPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim x As Integer
Dim y As Integer
    ' ****************************
    ' * Get the input bitmap data *
    ' ****************************
    ' Get a handle to the input bitmap.
    hbm = from_pict.Image
```

```
' See how big it is.
status = GetObject(hbm, BITMAP_SIZE, bm)
wid = bm.bmWidthBytes
hgt = bm.bmHeight

' Get the bits.
ReDim bytesin(0 To wid - 1, 0 To hgt - 1)
status = GetBitmapBits(hbm, wid * hgt, bytesin(0, 0))
ReDim bytesout(0 To wid - 1, 0 To hgt - 1)
' ********************
' * Apply the filter *
' ********************
' Get the current color values
hPal = from_pict.Picture.hPal
i = GetPaletteEntries(hPal, 0, 255, palentry(0))

' Do the erosion.
For x = Bound To wid - 1 - Bound
    ' If the operation has been canceled, stop.
    DoEvents
    If Not OperationRunning Then Exit For

    ' If we should show progress, do so.
    If show_progress Then
        status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
        to_pict.Refresh
    End If

    For y = Bound To hgt - 1 - Bound
        For i = -Bound To Bound
            For j = -Bound To Bound
                If palentry(bytesin(x + i, y + j)).peRed <> Kernel(i, j) Then Exit For
            Next j
            If j <= Bound Then Exit For
        Next i
        If j <= Bound Then
            c = (255 - Wgt) - bytesin(x, y)
        Else
            c = Wgt - bytesin(x, y)
        End If
        If c < 0 Then c = 0

        bytesout(x, y) = GetNearestPaletteIndex( _
            hPal, RGB(255 - c, 255 - c, 255 - c) + &H2000000)
    Next y
Next x
' ************************
' * Display the output *
' ************************
```

```
    status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
    to_pict.Refresh
End Sub
' ***************************************************
' Apply a morphological filter.
' Give the output pixel the value Wgt if the
' area's pixels match the kernel. Otherwise give
' it value 255 - Wgt.
' ***************************************************
Private Sub ApplyMorpho(from_pict As Object, to_pict As Object, show_progress As Boolean)
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim bytesin() As Byte
Dim bytesout() As Byte
Dim wid As Long
Dim hgt As Long
Dim i As Integer
Dim j As Integer
Dim c As Integer

Dim hPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim x As Integer
Dim y As Integer
    ' ******************************
    ' * Get the input bitmap data *
    ' ******************************
    ' Get a handle to the input bitmap.
    hbm = from_pict.Image

    ' See how big it is.
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight

    ' Get the bits.
    ReDim bytesin(0 To wid - 1, 0 To hgt - 1)
    status = GetBitmapBits(hbm, wid * hgt, bytesin(0, 0))
    ReDim bytesout(0 To wid - 1, 0 To hgt - 1)
    ' ********************
    ' * Apply the filter *
    ' ********************
    ' Get the current color values.
    hPal = from_pict.Picture.hPal
    i = GetPaletteEntries(hPal, 0, 255, palentry(0))

    ' Compute the new color values.
    For x = Bound To wid - 1 - Bound
```

```
        ' If the operation has been canceled, stop.
        DoEvents
        If Not OperationRunning Then Exit For

        ' If we should show progress, do so.
        If show_progress Then
            status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
            to_pict.Refresh
        End If

        For y = Bound To hgt - 1 - Bound
            For i = -Bound To Bound
                For j = -Bound To Bound
                    If palentry(bytesin(x + i, y + j)).peRed <> Kernel(i, j) Then Exit For
                Next j
                If j <= Bound Then Exit For
            Next i
            If j <= Bound Then
                c = 255 - Wgt
            Else
                c = Wgt
            End If
            bytesout(x, y) = GetNearestPaletteIndex( _
                hPal, RGB(c, c, c) + &H2000000)
        Next y
    Next x

    ' ************************
    ' * Display the output *
    ' ************************
    status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
    to_pict.Refresh
End Sub




' ***************************************************
' Apply an embossing filter. This is just like a
' normal filter except we add 127 to each new
' pixel value to give the image a gray background.
' ***************************************************
Private Sub ApplyEmboss(from_pict As Object, to_pict As Object, show_progress As Boolean)
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim bytesin() As Byte
Dim bytesout() As Byte
Dim wid As Long
```

```
Dim hgt As Long
Dim i As Integer
Dim j As Integer

Dim hPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim x As Integer
Dim y As Integer
Dim r As Long
Dim g As Long
Dim b As Long

    ' ****************************
    ' * Get the input bitmap data *
    ' ****************************
    ' Get a handle to the input bitmap.
    hbm = from_pict.Image

    ' See how big it is.
    status = GetObject(hbm, BITMAP_SIZE, bm)
    wid = bm.bmWidthBytes
    hgt = bm.bmHeight

    ' Get the bits.
    ReDim bytesin(0 To wid - 1, 0 To hgt - 1)
    status = GetBitmapBits(hbm, wid * hgt, bytesin(0, 0))
    ReDim bytesout(0 To wid - 1, 0 To hgt - 1)
    ' ********************
    ' * Apply the filter *
    ' ********************
    ' Get the current color values.
    hPal = from_pict.Picture.hPal
    i = GetPaletteEntries(hPal, 0, 255, palentry(0))

    ' Compute the new color values.
    For x = Bound To wid - 1 - Bound
        ' If the operation has been canceled, stop.
        DoEvents
        If Not OperationRunning Then Exit For

        ' If we should show progress, do so.
        If show_progress Then
            status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
            to_pict.Refresh
        End If
        For y = Bound To hgt - 1 - Bound
            r = 0
            g = 0
            b = 0
```

```
            For i = -Bound To Bound
                For j = -Bound To Bound
                    With palentry(bytesin(x + i, y + j))
                        r = r + Kernel(i, j) * .peRed
                        g = g + Kernel(i, j) * .peGreen
                        b = b + Kernel(i, j) * .peBlue
                    End With
                Next j
            Next i
            r = r / Wgt + 127
            g = g / Wgt + 127
            b = b / Wgt + 127
            If r < 0 Then r = 0
            If g < 0 Then g = 0
            If b < 0 Then b = 0
            bytesout(x, y) = GetNearestPaletteIndex( _
                hPal, RGB(r, g, b) + &H2000000)
        Next y
    Next x
    ' ************************
    ' * Display the output *
    ' ************************
    status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
    to_pict.Refresh
End Sub
' ***************************************************
' Apply a normal filter.
' ***************************************************
Private Sub ApplyNormal(from_pict As Object, to_pict As Object, show_progress As Boolean)
Dim bm As BITMAP
Dim hbm As Integer
Dim status As Long
Dim bytesin() As Byte
Dim bytesout() As Byte
Dim wid As Long
Dim hgt As Long
Dim i As Integer
Dim j As Integer

Dim hPal As Integer
Dim palentry(0 To 255) As PALETTEENTRY
Dim x As Integer
Dim y As Integer
Dim r As Long
Dim g As Long
Dim b As Long
    ' ******************************
    ' * Get the input bitmap data *
    ' ******************************
```

```
' Get a handle to the input bitmap.
hbm = from_pict.Image

' See how big it is.
status = GetObject(hbm, BITMAP_SIZE, bm)
wid = bm.bmWidthBytes
hgt = bm.bmHeight

' Get the bits.
ReDim bytesin(0 To wid - 1, 0 To hgt - 1)
status = GetBitmapBits(hbm, wid * hgt, bytesin(0, 0))
ReDim bytesout(0 To wid - 1, 0 To hgt - 1)

' ********************
' * Apply the filter *
' ********************
' Get the current color values.
hPal = from_pict.Picture.hPal
i = GetPaletteEntries(hPal, 0, 255, palentry(0))

' Compute the new color values.
For x = Bound To wid - 1 - Bound
    ' If the operation has been canceled, stop.
    DoEvents
    If Not OperationRunning Then Exit For

    ' If we should show progress, do so.
    If show_progress Then
        status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
        to_pict.Refresh
    End If

    For y = Bound To hgt - 1 - Bound
        r = 0
        g = 0
        b = 0
        For i = -Bound To Bound
            For j = -Bound To Bound
                With palentry(bytesin(x + i, y + j))
                    r = r + Kernel(i, j) * .peRed
                    g = g + Kernel(i, j) * .peGreen
                    b = b + Kernel(i, j) * .peBlue
                End With
            Next j
        Next i
        r = r / Wgt
        g = g / Wgt
        b = b / Wgt
        If r < 0 Then r = 0
```

```
            If g < 0 Then g = 0
            If b < 0 Then b = 0
            bytesout(x, y) = GetNearestPaletteIndex( _
                hPal, RGB(r, g, b) + &H2000000)
        Next y
    Next x

    ' ***********************
    ' * Display the output *
    ' ***********************
    status = SetBitmapBits(to_pict.Image, wid * hgt, bytesout(0, 0))
    to_pict.Refresh
End Sub



' ************************************************
' Display a message box giving the filter's
' dimensions
' ************************************************
Public Sub ShowVoting()
Dim numstr As String
Dim txt As String

    numstr = Format$(2 * Bound + 1)
    txt = "This is a " & numstr & "x" & numstr & _
        " voting filter."
    MsgBox txt, vbInformation, "Filter Values"
End Sub
' ************************************************
' Display a message box describing a morphological
' filter.
' ************************************************
Public Sub ShowMorpho()
Dim txt As String
Dim numstr As String
Dim maxlen As Integer
Dim x As Integer
Dim y As Integer

    ' See how long the biggest number is.
    maxlen = 0
    For y = -Bound To Bound
        For x = -Bound To Bound
            numstr = Format$(Kernel(x, y))
            If maxlen < Len(numstr) Then maxlen = Len(numstr)
        Next x
    Next y
```

```
    ' Build the message to display.
    For y = -Bound To Bound
        For x = -Bound To Bound
            numstr = Format$(Kernel(x, y))
            txt = txt & _
                Space$(maxlen - Len(numstr)) & _
                numstr & " "
        Next x
        txt = txt & vbCrLf
    Next y

    txt = txt & vbCrLf & _
        "If all value match:" & Str$(Wgt)
    txt = txt & vbCrLf & _
        "Otherwise:          " & Str$(255 - Wgt)

    MsgBox txt, vbInformation, "Filter Values"
End Sub
' ****************************************************
' Display a message box giving the filter's
' dimensions.
' ****************************************************
Public Sub ShowRank(rank_type As Integer)
Dim numstr As String
Dim typestr As String
Dim txt As String

    numstr = Format$(2 * Bound + 1)
    Select Case rank_type
        Case 0
            typestr = " minimum "
        Case 1
            typestr = " median "
        Case 2
            typestr = " maximum "
    End Select

    txt = "This is a " & numstr & "x" & _
        numstr & typestr & "filter."
    MsgBox txt, vbInformation, "Filter Values"
End Sub
' ****************************************************
' Display a message box showing the filter's
' components and weight.
' ****************************************************
Public Sub ShowNormal()
Dim txt As String
Dim numstr As String
Dim maxlen As Integer
```

```
Dim x As Integer
Dim y As Integer

    ' See how long the biggest number is.
    maxlen = 0
    For y = -Bound To Bound
        For x = -Bound To Bound
            numstr = Format$(Kernel(x, y), "0.00")
            If maxlen < Len(numstr) Then maxlen = Len(numstr)
        Next x
    Next y

    ' Build the message to display.
    For y = -Bound To Bound
        For x = -Bound To Bound
            numstr = Format$(Kernel(x, y), "0.00")
            txt = txt & _
                Space$(maxlen - Len(numstr)) & _
                numstr & " "
        Next x
        txt = txt & vbCrLf
    Next y
    txt = txt & vbCrLf & "Weight:" & Str$(Wgt)

    MsgBox txt, vbInformation, "Filter Values"
End Sub
' ***************************************************
' Display a message box describing the filter.
' ***************************************************
Public Sub ShowFilter()
    Select Case Kind
        Case FILTER_LOWPASS, FILTER_HIGHPASS, _
            FILTER_PREWITT, FILTER_LAPLACIAN, _
            FILTER_EMBOSS
            ShowNormal

        Case FILTER_RANK_MIN
            ShowRank 0

        Case FILTER_RANK_MEDIAN
            ShowRank 1

        Case FILTER_RANK_MAX
            ShowRank 2

        Case FILTER_VOTING
            ShowVoting

        Case FILTER_MORPHO
```

```
            ShowMorpho

        Case FILTER_ERODE_OUTLINE
            ShowErodeOutline

        Case FILTER_DILATE_OUTLINE
            ShowDilateOutline
    End Select
End Sub
' ****************************************************
' Initialize the filter based on its name.
' ****************************************************
Sub InitializeFilter(filtername As String)
    Select Case filtername
        Case "Average 3x3"
            InitializeAverage 3

        Case "Low Pass 3x3"
            InitializeLowPass 3
        Case "Low Pass 5x5"
            InitializeLowPass 5
        Case "Low Pass 7x7"
            InitializeLowPass 7

        Case "High Pass 1"
            InitializeHighPass 1
        Case "High Pass 2"
            InitializeHighPass 2
        Case "High Pass 3"
            InitializeHighPass 3
        Case "High Pass 4"
            InitializeHighPass 4

        Case "Prewitt Up"
            InitializePrewitt FILTER_UP
        Case "Prewitt Up-Right"
            InitializePrewitt FILTER_UP_RIGHT
        Case "Prewitt Right"
            InitializePrewitt FILTER_RIGHT
        Case "Prewitt Down-Right"
            InitializePrewitt FILTER_DOWN_RIGHT
        Case "Prewitt Down"
            InitializePrewitt FILTER_DOWN
        Case "Prewitt Down-Left"
            InitializePrewitt FILTER_DOWN_LEFT
        Case "Prewitt Left"
            InitializePrewitt FILTER_LEFT
        Case "Prewitt Up-Left"
            InitializePrewitt FILTER_UP_LEFT
```

```
    Case "Laplacian 1"
        InitializeLaplacian 1
    Case "Laplacian 2"
        InitializeLaplacian 2

    Case "Minimum 3x3"
        Kind = FILTER_RANK_MIN
        Bound = 1
    Case "Median 3x3"
        Kind = FILTER_RANK_MEDIAN
        Bound = 1
    Case "Maximum 3x3"
        Kind = FILTER_RANK_MAX
        Bound = 1

    Case "Voting 3x3"
        InitializeVoting 3

    Case "Emboss Up"
        InitializeEmboss FILTER_UP
    Case "Emboss Up-Right"
        InitializeEmboss FILTER_UP_RIGHT
    Case "Emboss Right"
        InitializeEmboss FILTER_RIGHT
    Case "Emboss Down-Right"
        InitializeEmboss FILTER_DOWN_RIGHT
    Case "Emboss Down"
        InitializeEmboss FILTER_DOWN
    Case "Emboss Down-Left"
        InitializeEmboss FILTER_DOWN_LEFT
    Case "Emboss Left"
        InitializeEmboss FILTER_LEFT
    Case "Emboss Up-Left"
        InitializeEmboss FILTER_UP_LEFT

    ' Morphological filters.
    Case "Erosion"
        InitializeMorpho FILTER_EROSION
    Case "Dilation"
        InitializeMorpho FILTER_DILATION

    ' Outlines.
    Case "Erosion Outline"
        InitializeErodeOutline
    Case "Dilation Outline"
        InitializeDilateOutline
  End Select
End Sub
```

```
' ****************************************************
' Apply the filter to an array of bits.
' ****************************************************
Public Sub ApplyFilter(from_pict As Object, to_pict As Object, show_progress As Boolean)
    Select Case Kind
        Case FILTER_NONE
            Beep
            MsgBox "This filter is undefined.", vbExclamation

        Case FILTER_LOWPASS, FILTER_HIGHPASS, _
             FILTER_PREWITT, FILTER_LAPLACIAN
            ApplyNormal from_pict, to_pict, show_progress

        Case FILTER_EMBOSS
            ApplyEmboss from_pict, to_pict, show_progress

        Case FILTER_RANK_MEDIAN, FILTER_RANK_MIN, _
             FILTER_RANK_MAX
            ApplyRank from_pict, to_pict, show_progress

        Case FILTER_VOTING
            ApplyVoting from_pict, to_pict, show_progress

        Case FILTER_MORPHO
            ApplyMorpho from_pict, to_pict, show_progress

        Case FILTER_ERODE_OUTLINE
            ApplyErodeOutline from_pict, to_pict, show_progress

        Case FILTER_DILATE_OUTLINE
            ApplyDilateOutline from_pict, to_pict, show_progress

    End Select
End Sub
' ****************************************************
' Initialize a high pass (sharpening) filter.
' ****************************************************
Public Sub InitializeHighPass(high_kind As Integer)
Dim r As Integer
Dim c As Integer

    Kind = FILTER_HIGHPASS
    Bound = 1
    ReDim Kernel(-Bound To Bound, -Bound To Bound)

    Select Case high_kind
        Case 1
```

```
            For r = -Bound To Bound
                For c = -Bound To Bound
                    Kernel(r, c) = -1
                Next c
            Next r
            Kernel(0, 0) = 9
            Wgt = 1

        Case 2
            Kernel(-1, -1) = 0
            Kernel(-1, 0) = -1
            Kernel(-1, 1) = 0
            Kernel(0, -1) = -1
            Kernel(0, 0) = 5
            Kernel(0, 1) = -1
            Kernel(1, -1) = 0
            Kernel(1, 0) = -1
            Kernel(1, 1) = 0
            Wgt = 1

        Case 3
            Kernel(-1, -1) = 1
            Kernel(-1, 0) = -2
            Kernel(-1, 1) = 1
            Kernel(0, -1) = -2
            Kernel(0, 0) = 5
            Kernel(0, 1) = -2
            Kernel(1, -1) = 1
            Kernel(1, 0) = -2
            Kernel(1, 1) = 1
            Wgt = 1

        Case 4
            For r = -Bound To Bound
                For c = -Bound To Bound
                    Kernel(r, c) = -1
                Next c
            Next r
            Kernel(0, 0) = 15
            Wgt = Kernel(0, 0) - 8

        Case Else
            ' Flag the filter as uninitialized.
            Kind = FILTER_NONE
            Wgt = 1

    End Select
End Sub
```

```
' ****************************************************
' Initialize a voting filter.
' ****************************************************
Public Sub InitializeVoting(size As Integer)
    Kind = FILTER_VOTING
    Bound = size \ 2
End Sub

' ****************************************************
' Initialize an averaging (blurring) filter.
' ****************************************************
Public Sub InitializeAverage(size As Integer)
Dim r As Integer
Dim c As Integer

    Kind = FILTER_LOWPASS
    Bound = size \ 2
    ReDim Kernel(-Bound To Bound, -Bound To Bound)

    For r = -Bound To Bound
        For c = -Bound To Bound
            Kernel(r, c) = 1
        Next c
    Next r
    Wgt = (2 * Bound + 1) * (2 * Bound + 1)
End Sub

' ****************************************************
' Initialize a 3x3 Laplacian filter.
' ****************************************************
Public Sub InitializeLaplacian(laplacian_type As Integer)
Dim r As Integer
Dim c As Integer

    Kind = FILTER_LAPLACIAN
    Bound = 1
    ReDim Kernel(-Bound To Bound, -Bound To Bound)

    Select Case laplacian_type
        Case 1
            For r = -Bound To Bound
                For c = -Bound To Bound
                    Kernel(r, c) = -1
                Next c
            Next r
            Kernel(0, 0) = 8

        Case 2
            Kernel(-1, -1) = 0
```

```
            Kernel(0, -1) = -1
            Kernel(1, -1) = 0
            Kernel(-1, 0) = -1
            Kernel(0, 0) = 4
            Kernel(1, 0) = -1
            Kernel(-1, 1) = 0
            Kernel(0, 1) = -1
            Kernel(1, 1) = 0

    End Select
    Wgt = 1
End Sub

' ****************************************************
' Initialize a 3x3 dilation filter for use when
' creating an outline using dilation.
' ****************************************************
Public Sub InitializeDilateOutline()
    InitializeMorpho FILTER_DILATION
    Kind = FILTER_DILATE_OUTLINE
End Sub
' ****************************************************
' Initialize a 3x3 erosion filter for use when
' creating an outline using erosion.
' ****************************************************
Public Sub InitializeErodeOutline()
    InitializeMorpho FILTER_EROSION
    Kind = FILTER_ERODE_OUTLINE
End Sub

' ****************************************************
' Initialize a 3x3 morphological filter.
' ****************************************************
Public Sub InitializeMorpho(morpho_type As Integer)
Dim r As Integer
Dim c As Integer

    Kind = FILTER_MORPHO
    Bound = 1
    ReDim Kernel(-Bound To Bound, -Bound To Bound)

    Select Case morpho_type
        Case FILTER_EROSION
            For r = -Bound To Bound
                For c = -Bound To Bound
                    Kernel(r, c) = 255
                Next c
            Next r
            Wgt = 255
```

```
        Case FILTER_DILATION
            For r = -Bound To Bound
                For c = -Bound To Bound
                    Kernel(r, c) = 0
                Next c
            Next r
            Wgt = 0
    End Select
End Sub



' ************************************************
' Initialize a 3x3 embossing filter.
' ************************************************
Public Sub InitializeEmboss(emboss_type As Integer)
Dim r As Integer
Dim c As Integer

    Kind = FILTER_EMBOSS
    Bound = 1
    ReDim Kernel(-Bound To Bound, -Bound To Bound)

    For r = -Bound To Bound
        For c = -Bound To Bound
            Kernel(r, c) = 0
        Next c
    Next r

    Select Case emboss_type
        Case FILTER_UP
            Kernel(0, -1) = 1: Kernel(0, 1) = -1
        Case FILTER_UP_RIGHT
            Kernel(-1, 1) = -1: Kernel(1, -1) = 1
        Case FILTER_RIGHT
            Kernel(1, 0) = 1: Kernel(-1, 0) = -1
        Case FILTER_DOWN_RIGHT
            Kernel(-1, -1) = -1: Kernel(1, 1) = 1
        Case FILTER_DOWN
            Kernel(0, -1) = -1: Kernel(0, 1) = 1
        Case FILTER_DOWN_LEFT
            Kernel(-1, 1) = 1: Kernel(1, -1) = -1
        Case FILTER_LEFT
            Kernel(1, 0) = -1: Kernel(-1, 0) = 1
        Case FILTER_UP_LEFT
            Kernel(-1, -1) = 1: Kernel(1, 1) = -1
    End Select
    Wgt = 1
```

```
End Sub
' ****************************************************
' Initialize a 3x3 Prewitt filter
' ****************************************************
Public Sub InitializePrewitt(prewitt_type As Integer)
Dim r As Integer
Dim c As Integer

    Kind = FILTER_PREWITT
    Bound = 1
    ReDim Kernel(-Bound To Bound, -Bound To Bound)

    For r = -Bound To Bound
        For c = -Bound To Bound
            Kernel(r, c) = 1
        Next c
    Next r
    Kernel(0, 0) = -2

    Select Case prewitt_type
        Case FILTER_UP
            Kernel(-1, 1) = -1: Kernel(0, 1) = -1: Kernel(1, 1) = -1
        Case FILTER_UP_RIGHT
            Kernel(-1, 0) = -1: Kernel(-1, 1) = -1: Kernel(0, 1) = -1
        Case FILTER_RIGHT
            Kernel(-1, -1) = -1: Kernel(-1, 0) = -1: Kernel(-1, 1) = -1
        Case FILTER_DOWN_RIGHT
            Kernel(-1, -1) = -1: Kernel(-1, 0) = -1: Kernel(0, -1) = -1
        Case FILTER_DOWN
            Kernel(-1, -1) = -1: Kernel(0, -1) = -1: Kernel(1, -1) = -1
        Case FILTER_DOWN_LEFT
            Kernel(0, -1) = -1: Kernel(1, -1) = -1: Kernel(1, 0) = -1
        Case FILTER_LEFT
            Kernel(1, -1) = -1: Kernel(1, 0) = -1: Kernel(1, 1) = -1
        Case FILTER_UP_LEFT
            Kernel(0, 1) = -1: Kernel(1, 0) = -1: Kernel(1, 1) = -1
    End Select
    Wgt = 1
End Sub



' ****************************************************
' Initialize a low pass (blurring) filter
' ****************************************************
Public Sub InitializeLowPass(size As Integer)
Dim r As Integer
Dim c As Integer
Dim vr As Integer
```

```
    Kind = FILTER_LOWPASS
    Bound = size \ 2
    ReDim Kernel(-Bound To Bound, -Bound To Bound)

    For r = -Bound To Bound
        vr = Bound + 1 - Abs(r)
        For c = -Bound To Bound
            Kernel(r, c) = vr * (Bound + 1 - Abs(c))
            Wgt = Wgt + Kernel(r, c)
        Next c
    Next r
End Sub

' ***************************************************
' Flag the filter as uninitialized.
' ***************************************************
Private Sub Class_Initialize()
    Kind = FILTER_NONE
End Sub
```

ไฟล์ filter1.bas

```
Attribute VB_Name = "Filters"
Option Explicit

Type PALETTEENTRY
    peRed As Byte
    peGreen As Byte
    peBlue As Byte
    peFlags As Byte
End Type
Public Const PC_EXPLICIT = &H2      ' Match to system palette index.
Public Const PC_NOCOLLAPSE = &H4    ' Do not match color existing entries.

' GetDeviceCaps constants.
Global Const RASTERCAPS = 38    ' Raster device capabilities.
Global Const RC_PALETTE = &H100 ' Has palettes.
Global Const NUMRESERVED = 106  ' # reserved entries in palette.
Global Const SIZEPALETTE = 104  ' Size of system palette.

#If Win32 Then  ' 32-bit VB.
    Type BITMAP ' 24 bytes
        bmType As Long
        bmWidth As Long
        bmHeight As Long
        bmWidthBytes As Long
        bmPlanes As Integer
        bmBitsPixel As Integer
        bmBits As Long
    End Type
    Global Const BITMAP_SIZE = 24
    Declare Function GetDeviceCaps Lib "gdi32" (ByVal hdc As Integer, ByVal nIndex As Integer) As Integer
    Declare Function ResizePalette Lib "gdi32" (ByVal hPalette As Integer, ByVal nNumEntries As Integer) As Integer
    Declare Function SetPaletteEntries Lib "gdi32" (ByVal hPalette As Integer, ByVal wStartIndex As Integer, ByVal wNumEntries As Integer,
lpPaletteEntries As PALETTEENTRY) As Integer
    Declare Function GetPaletteEntries Lib "gdi32" (ByVal hPalette As Integer, ByVal wStartIndex As Integer, ByVal wNumEntries As Integer,
lpPaletteEntries As PALETTEENTRY) As Integer
    Declare Function GetSystemPaletteEntries Lib "gdi32" (ByVal hdc As Integer, ByVal wStartIndex As Integer, ByVal wNumEntries As Integer,
lpPaletteEntries As PALETTEENTRY) As Integer
    Declare Function RealizePalette Lib "gdi32" (ByVal hdc As Long) As Long
    Declare Function GetBitmapBits Lib "gdi32" (ByVal hBitmap As Integer, ByVal dwCount As Long, lpBits As Any) As Long
    Declare Function SetBitmapBits Lib "gdi32" (ByVal hBitmap As Integer, ByVal dwCount As Long, lpBits As Any) As Long
    Declare Function GetObject Lib "gdi32" Alias "GetObjectA" (ByVal hObject As Long, ByVal nCount As Long, lpObject As Any) As Long
    Declare Function GetNearestPaletteIndex Lib "gdi32" (ByVal hPalette As Integer, ByVal crColor As Long) As Integer
#Else           ' 16-bit VB.
    Type BITMAP ' 14 bytes
        bmType As Integer
        bmWidth As Integer
        bmHeight As Integer
```

```
        bmWidthBytes As Integer
        bmPlanes As String * 1
        bmBitsPixel As String * 1
        bmBits As Long
    End Type
    Global Const BITMAP_SIZE = 14
    Declare Function GetDeviceCaps Lib "GDI" (ByVal hdc As Integer, ByVal nIndex As Integer) As Integer
    Declare Function ResizePalette Lib "GDI" (ByVal hPalette As Integer, ByVal nNumEntries As Integer) As Integer
    Declare Function SetPaletteEntries Lib "GDI" (ByVal hPalette As Integer, ByVal wStartIndex As Integer, ByVal wNumEntries As Integer,
lpPaletteEntries As PALETTEENTRY) As Integer
    Declare Function GetPaletteEntries Lib "GDI" (ByVal hPalette As Integer, ByVal wStartIndex As Integer, ByVal wNumEntries As Integer,
lpPaletteEntries As PALETTEENTRY) As Integer
    Declare Function GetSystemPaletteEntries Lib "GDI" (ByVal hdc As Integer, ByVal wStartIndex As Integer, ByVal wNumEntries As Integer,
lpPaletteEntries As PALETTEENTRY) As Integer
    Declare Function RealizePalette Lib "User" (ByVal hdc As Integer) As Integer
    Declare Function GetBitmapBits Lib "GDI" (ByVal hBitmap As Integer, ByVal dwCount As Long, lpBits As Any) As Long
    Declare Function SetBitmapBits Lib "GDI" (ByVal hBitmap As Integer, ByVal dwCount As Long, lpBits As Any) As Long
    Declare Function GetObject Lib "GDI" (ByVal hObject As Integer, ByVal nCount As Integer, lpObject As Any) As Integer
    Declare Function GetNearestPaletteIndex Lib "GDI" (ByVal hPalette As Integer, ByVal crColor As Long) As Integer
#End If

Global OperationRunning As Boolean
```

ไฟล์ varglob.h

```
/*****************************************************************/
/*                          VARGLOB                              */
/*****************************************************************/

#ifndef _VARGLOB_H
#define _VARGLOB_H

#ifdef DATA_HERE
#define DEMOEXTERN
#else
#define DEMOEXTERN extern
#endif

/*****************************************************************/
/*                    constants of the program                   */
/*****************************************************************/

#define DEMO_BADFILE1 {MessageBox(NULL,"Unable to read the file.\nThe file format is not right or the file is damaged\n","Error !",MB_OK);
_exit(-1);}
#define DEMO_BADFILE2 {MessageBox(NULL,"Not enough memory.\n","Error !",MB_OK);_exit(-1);}

#define MAX_PUNTI 4000          /* maximum number of points of the contour      */
#define DIMMAT_IN 352           /* dimension of the matrix of the input picture */

#define FALSE 0
#define TRUE 1

/*****************************************************************/
/*                    types of  variables of the program         */
/*****************************************************************/

typedef unsigned char BYTE;

typedef unsigned short WORD;

typedef unsigned long DWORD;

typedef struct tagBITMAPFILEHEADER1{
        WORD    bfType;
        DWORD   bfSize;
        WORD    bfReserved1;
        WORD    bfReserved2;
        DWORD   bfOffBits;
}BITMAPFILEHEADER1;

typedef struct tagRGBQUAD1{
```

```
        BYTE    rgbBlue;
        BYTE    rgbGreen;
        BYTE    rgbRed;
        BYTE    rgbReserved;
}RGBQUAD1;

typedef struct tagfBITMAPINFOHEADER{
        DWORD   biSize;
        DWORD   biWidth;
        DWORD   biHeight;
        WORD    biPlanes;
        WORD    biBitCount;
        DWORD   biCompression;
        DWORD   biSizeImage;
        DWORD   biXPelsPerMeter;
        DWORD   biYPelsPerMeter;
        DWORD   biClrUsed;
        DWORD   biXClrImportant;
}BITMAPINFOHEADER1;

typedef struct tagfBITMAPINFO{
        BITMAPINFOHEADER1       bmiHeader;
        RGBQUAD1                bmiColors[1];
}BITMAPINFO1;

typedef struct nodo{
    int ver;
    struct nodo *ad;
}NODO;

typedef struct VETT{
    int asc;
    int ord;
    int mis;
}VETT;

typedef void(*SF)(void);

/*****************************************************************/
/*                      global variables of the program          */
/*****************************************************************/

DEMOEXTERN unsigned char huge mat_in[DIMMAT_IN][DIMMAT_IN];  /*          matrix of the input picture */
DEMOEXTERN VETT huge bordo[MAX_PUNTI];/*          array that contains the co-ordinates of the  contour of the image  */
DEMOEXTERN int xb,yb;                                       /*          co-ordinates of barycentre                 */
DEMOEXTERN int cont;                                        /*          number of points of the image contour */

#endif
```

ไฟล์ include.h

```
/*****************************************************************/
/*                        INCLUDE                                */
/*****************************************************************/
/*                        list of subroutine                     */
/*****************************************************************/

extern int far pascal readbitmap();
extern int far pascal edge();
```

ประวัติผู้เขียน

นายอภิชาติ สระวาสี เกิดวันที่ 12 กันยายน พ.ศ. 2505 ที่อำเภอดุสิต กรุงเทพมหานคร สำเร็จปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมเครื่องกล จากจุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2529 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2537 ปัจจุบันเป็นพนักงานรัฐวิสาหกิจ ตำแหน่ง วิศวกรระดับ 6 ทำหน้าที่วิศวกรประจำแผนกทดสอบเครื่องจักร กองทดสอบเครื่องกล ฝ่ายบำรุงรักษาเครื่องกล การไฟฟ้าฝ่ายผลิตแห่งประเทศไทย