```fortran
!RedGB3D.F95
!2016.09.12.1705cdt JMS: Identify one corresponding pixel in each image,
!                           align them, and discard the non-overlaps.
!2016.09.07.1010cdt JMS: The left side of both images- truncation sign error
!2015.09.26.1200cdt JMS: Added Pixels Convergence Adjustment
!2015.09.21.1120cdt JMS

!Module RedGB3DDec
!         - Key program variables:
!Program RedGB3D
!         - Imports: a full-color or gray-scale side-by-side 3D .bmp image
!         & exports: a Red-GreenBlue 3D .bmp image.
!Subroutine ReadBmpImageHeader(I)
!         - Reads a bitmap's BH header.
!Subroutine PrintBmpImageHeader(BH)-
!         - Reads a bitmap's BH header.
!Subroutine Fdate20(DaTime)
!         - Acquire the current date & Time.

!---------------------------------------------------------------------------7 9
!---------------------------------------------------------------------------7 9
Module RedGB3DDec
!2016.09.12.1705cdt JMS: Identify one corresponding pixel in each image,
!                           align them, and discard the non-overlaps.
!2016.09.09.1000cdt JMS: Use a corresponding pixel for alignment.
!2015.09.26.1130cdt JMS: Added Pixels Convergence Adjustment
!2015.09.21.1120cdt JMS- Key program variables:
!                      - Traveler2/Athlon64/WinXPPro/APF9.0
!                        System ID/  CPU   /OS      /Compiler
 Implicit none

!Ref: Graphics File Formats, Kay & Levine,  ISBN 0-8306-3059-7  P.115-119
 type      ::BmpHdr                        !Bitmap Header  (54 bytes)
  character  :: BM*2                 !    BM   mandatory
  integer*4  :: nSizeTot             !   54+nClrUsed*4
                                    !+(((nWidth*(nBitsPP/8)+3)/4)*4)*nHeight *
  integer*2  :: nReserv1,nReserv2   !   0, 0 not used
  integer*4  :: nOffBit,nSizeH,nWidth,nHeight ! 54,40,___,___(pixels,pixels)
  integer*2  :: nPlanes,nBitsPP     !   1,___  (24:TC,  8:Grey Scales)
  integer*4  :: nCompres,nSizeC     !   0,___  ...some ortho's use in error.
  real*4     :: XPpM,YPpM           !   ___,___ scaling (USGS float)
  integer*4  :: nClrUsed,nClrImpo   !   ___, 0 ( 0:TC,256:Grey Scales)
 end type BmpHdr;
     type(BmpHdr):: BH

!        *** For 8-bit-grayscale images & 24-bit-color images: ***
!
!After the 54 bytes of header information...
!  rows of pixel data follow, starting at the lower left of the image.
!  Each pixel's color is in Blue/Green/Red order with one byte per color.
!  Each color is in the range [0,255]- which is 8 bits = one byte. 3 bytes/pix.
!  Grayscale pixels are       [0,255]- but only 1 byte/pix. is needed & used.

!Bitmap images pad each row of pixels to the next 4-byte boundary.
!If the number of pixels in a row is divisible by 4, then there's no padding.

!Since 54/3 = 18 is a round number, the merging of two unpadded 24-bit-color
!images with the same (BH) header information can be accomplished by:
!       overwriting every 3rd byte of the left image (the red color for 55+)
!             onto the corresponding byte of the right image.
!         Thus, the right image will morph into the -RedGB3D.bmp image.

!The program below converts side-by-side 3D .bmp images to -RedGB3D.bmp images.

!Reading and writing entire images from/to disk saves a lot of runtime.

 type    ::ImageRec
  integer*4  ::N            ! Ir(N)
  integer*4  ::EyeOffset    ! =-1:right/left; = 0:don't know; = 1:left/right
  character  ::BmpFileName*60
  integer*4  ::BytesPerPixel!number of characters per pixel
  integer*4  ::PixelsPerRow
  integer*4  ::BytesPerRow
  integer*4  ::PadBytes     !extra bytes to end each row on a 4-byte boundary
  integer*4  ::BytesPerRowPadded
  integer*4  ::PixelHeight
  integer*4  ::ImageBytesTotalPadded!=PixHeight*(BytesPerRow+PadBytes)
  type(BmpHdr)::BH          !The image header information - 54 bytes.
 end type  ImageRec; type(ImageRec),public::I1,I2
                            !               I1.    is the input  image
                            !                  I2. is the output image

  integer*4  ::LeftPixel( 2)!Pixel to correspond- Left                      >=0
  integer*4  ::RightPixel(2)!                   - Right                       "
  integer*4  ::LeftNet(   2)!Image pixel shift  - Left                        "
  integer*4  ::RightNet(  2)!                   - Right                       "

  integer*4  ::   PCA     !Pixels Convergence Adjustment-
  integer*4  ::absPCA     !                             - absolute value
                          !The image width is narrowed by abs(PCA)
       !The left of one image & the right of the other image are truncated.
  integer*4  ::   PVA     !Pixels Vertical Adjustment   -
  integer*4  ::absPVA     !                             - absolute value
                          !The image height is narrowed by abs(PVA)
       !The bottom of one image & the top of the other image are truncated.

  character*1,allocatable::BmpIn(  :,:) !(I1.BytesPerRowPadded,I1.PixelHeight)
  character*1,allocatable::BmpOut( :,:) !(I2.BytesPerRowPadded,I2.PixelHeight)

  logical  ::Exists        !File existance flag
  integer*4::iAlloc        !Memory allocation error flag
  character::DaTime*20      !Current date & time.
  integer*4::iUnitR=12      !Fortran Unit number for- reading- file(s)
  integer*4::iUnitW=14      !                       - writing-
  integer*4::iP    =6      !                        - the screen

  integer*4::iCharL
  character:: CharL*1
  equivalence (icharL,CharL)

  character::ProgramInfo(56)*75
  data ProgramInfo / &
!"       RedGB3D.exe   version 0.5    2015.09.27   Jeff Setterholm          "&
 "       RedGB3D.exe   version 0.6    2016.09.12   Jeff Setterholm          "&
,"                      ~ 'rev. A'                 jeff.setterholm@gmail.com"&
,"                                                                          "&
," On the web: http://ftp.setterholm.com/Fortran/RedGB3D/RedGB3D.exf        "&
,"                                                 change to:   ^.exe to run"&
,"                Initialization file:  /Fortran/RedGB3D/RedGB3D.ini        "&
,"                     Source code   :  /Fortran/RedGB3D/RedGB3D.f95        "&
,"                                                                          "&
," Your text file named `RedGB3D.ini` initializes this program by providing "&
," five pieces of information:                                              "&
,"                                                                          "&
," 1. Your input .bmp image file name - a side-by-side stereo pair          "&
,"    which can be either 24-bit color or 8-bit grayscale.                  "&
,"                                                                          "&
," 2. a flag = -1 for a Right/Left ( =   crossed-eye) input image           "&
,"           = +1 for a Left/Right (e.g. stereoscope) input image           "&
,"                                                                          "&
," 3. The name for the resulting `-REDGB3d.bmp` image'                      "&
,"                                                                          "&
," 4. The `corresponding pixels' in- the  left image: horizontal & vertical "&
,' 5.                              - the right image:     "      &    "     '&
,"    Using MSPaint's convention: the upper left pixel = 0 0.               "&
,"    The program operates on two differences:                              "&
,"      `Adjust(1)`= Left image horizontal pix.- right image horizontal pix."&
,"         >0 moves convergence depth: farther into the scene               "&
,"            removes pixel columns from:  left image- left  edge           "&
,"                                & from: right image- right edge           "&
,"         <0 moves convergence depth:  nearer  in  the scene               "&
,"            removes pixel columns from:  left image- right edge           "&
,"                                & from: right image- left  edge           "&
,"      `Adjust(2)`= Left image  vertical  pix.- right image  vertical  pix."&
,"         >0 removes pixel   rows  from:  left image-  top   edge          "&
,"                                & from: right image- bottom edge          "&
,"         <0 removes pixel   rows  from:  left image- bottom edge          "&
,"                                & from: right image-  top   edge          "&
,"    The resulting images coincide at the chosen pixel.                    "&
,"                                                                          "&
,"    In most cases... rotational mismatches will remain;                   "&
,"                     this algorithm is only a simple approximation.       "&
,"    Image warping & lens distortion corrections are usually needed        "&
,"      to achieve precise inter-image stereo cohesiveness...               "&
,"      unless these corrections are already incorporated the 3D imager.    "&
,"                                                                          "&
,"Example:                                                                  "&
,'"qVP.bmp"C              <- Your input .bmp image filename                 '&
,'-1                      <- -1 for Right/Left ; +1 for Left/Right          '&
,'"qVP-RedGB3D.bmp"C      <- Your name for the output -RedGB3d.bmp image    '&
,'    0    0              <- LeftPixel( 2)   hor.>=0 & vert.>=0             '&
,'    0    0              <- RightPixel(2)       "          "               '&
,"                                                                          "&
,"              This software is experimental & has NO warranties.          "&
,"                In particular, input error checking is minimal.           "&
,"              You`re welcome to improve the source code yourself.         "&
,"                                                                          "&
,"                    Use the program ONLY at your own risk.                "&
,"                                                                          "/
End Module RedGB3DDec
!------------------------------------------------------------------------------

Program RedGB3D !--------------------------------------------------------------

!2016.09.12.1705cdt JMS: Identify one corresponding pixel in each image,
!                           align them, and discard the non-overlaps.
!2015.09.26.1130cdt JMS: Added Pixels Convergence Adjustment
!2015.09.27.0645cdt JMS- Added convergence adjustment. &
!2015.09.21.1120cdt JMS- Imports: a full-color or gray-scale
!                                   side-by-side 3D .bmp image
!                      & exports: a Red-GreenBlue 3D .bmp image.
!-----                 - Traveler2/Athlon64/WinXPPro/APF9.0
 use RedGB3DDec
 implicit none
  integer*4::  Icontinue,LES,RES,PixMult
  integer*4  ::i,n,m,iZero
  !----------                                               End of declarations
                              !On-screen TimeStamp...
                               call Fdate20(DaTime);write(6,"(59x,a20)") DaTime
   write(6,"(36(/a75))") ProgramInfo

                               call Fdate20(DaTime);write(6,"(59x,a20)") DaTime

   !Identify the Image file that you want to convert to RedGB3D:
   open(unit=iUnitR,File='RedGB3D.ini',action='read')
     read(   iUnitR,*) I1.BmpFileName                !Input image
         write(6,*) 'Input  file: ', I1.BmpFileName
     read(   iUnitR,*) I1.EyeOffset
                   if(I1.EyeOffset>1) I1.EyeOffset= 1  !Left/Right
                   if(I1.EyeOffset<1) I1.EyeOffset=-1  !Right/Left
         write(6,*) 'Eye  Offset: ',   I1.EyeOffset
     read(   iUnitR,*) I2.BmpFileName               !Output image
         write(6,*) 'Output file: ', I2.BmpFileName
     read(   iUnitR,*) LeftPixel
         write(6,"('Left  convergence Pixel: ',2i6 )")  LeftPixel
     read(   iUnitR,*) RightPixel
         write(6,"('Right convergence Pixel: ',2i6/)")  RightPixel
   close(    iUnitR)

   write(6,"('Press 0 to zero the offsets  -or-  Press 1 to use them:',\)")
```

```fortran
read( 5,*) IZero
LeftPixel = LeftPixel*iZero
RightPixel=RightPixel*iZero


I1.N=1; i2.N=2 !Set the indices of both ImageRec's
do i=1,2
  LeftNet(i)= LeftPixel(i)-RightPixel(i); if( LeftNet(i).lt.0)  LeftNet(i)=0
 RightNet(i)=RightPixel(i)- LeftPixel(i); if(RightNet(i).lt.0) RightNet(i)=0
enddo!i
PCA=LeftNet(1)-RightNet(1); absPCA=iabs(PCA)
PVA=LeftNet(2)-RightNet(2); absPVA=iabs(PVA)
      write(6,"('Left    net       Pixel: ',2i6 )")  LeftNet
      write(6,"('Right   net       Pixel: ',2i6 )")  RightNet
      write(6,"('       net convergence: ',2i6/)")  PCA,PVA

call ReadBmpImageHeader(I1)

!Allocate memory for the image:
allocate(BmpIn( I1.BytesPerRowPadded, I1.PixelHeight ),stat=iAlloc)
if(iAlloc.ne.0) then;
  pause 'BmpIn(*:*) allocation error. Halt.'; stop
endif!(iAlloc<>0)

!Import the image:
open(unit=iUnitR,file=I1.BmpFileName,form='binary',action='read')
  read(   iUnitR)     I1.BH, BmpIn
close(    iUnitR)

!Size the output image:
I2.PixelHeight       =           I1.PixelHeight   -absPVA
I2.PixelsPerRow      =           I1.PixelsPerRow/2-absPCA
I2.BytesPerPixel     =              3
I2.BytesPerRow       =           I2.PixelsPerRow   &
                               * I2.BytesPerPixel
I2.PadBytes          = mod(4-mod(I2.BytesPerRow,4),4)
I2.BytesPerRowPadded =           I2.BytesPerRow    &
                               + I2.PadBytes
!Modify the output bitmap header:
I2.BH                =           I1.BH !Copy input's BH to the output's BH.
I2.BH.nHeight        =           I2.PixelHeight
I2.BH.nSizeTot       =       54+ I2.BytesPerRowPadded*I2.PixelHeight
I2.BH.nWidth         =           I2.PixelsPerRow
I2.BH.nBitsPP        =       24

write(6,"(/'Output image:')")
call PrintBmpImageHeader(I2.BH)
write(6,"(/'I',i1,'.PixelsPerRow     =',i8)") I2.N,I2.PixelsPerRow
write(6,"( '  ',  ' PixelsConvAdj    =',i8)") PCA
write(6,"( '  ',  ' PixelsVertAdj    =',i8)") PVA
write(6,"( 'I',i1,'.BytesPerPix      =',i8)") I2.N,I2.BytesPerPixel
write(6,"( 'I',i1,'.BytesPerRow      =',i8)") I2.N,I2.BytesPerRow
write(6,"( 'I',i1,'.PadBytes         =',i8)") I2.N,I2.PadBytes
write(6,"( 'I',i1,'.BytesPerRowPadded=',i8)") I2.N,I2.BytesPerRowPadded

!Allocate memory for the output image - BmpOut(*:*):
allocate(BmpOut(I2.BytesPerRowPadded,I2.PixelHeight),stat=iAlloc)
if(iAlloc.ne.0) then
  pause 'BmpOut(*:*)  image allocation error. Halt.'; stop
endif!(iAlloc<>0)
BmpOut=char(0)  !...which clears the end-of-row-padding, if any.

!Determine the byte shifts & multiplier
select case(I1.EyeOffset)
 case(-1) !Right/left    (crossed-eyes)    input:
  LES = I1.BytesPerRow/2 !Left  Eye Shift
  RES=     0             !Right Eye Shift
 case( 1) !Left/right (e.g.stereoscope) input:
  LES =    0             !Left  Eye Shift
  RES = I1.BytesPerRow/2 !Left  Eye Shift
```

```
   end select
   PixMult=I1.BytesPerPixel !pixel size multiplier

   if(PixMult.eq.1) then
   do   n= 1,I2.PixelHeight
     do m= 0,I2.PixelsPerRow-1
       BmpOut(m*3+1,n)=BmpIn((m+RightNet(1))*PixMult+RES,n+ LeftNet(2))!Blue
       BmpOut(m*3+2,n)=BmpIn((m+RightNet(1))*PixMult+RES,n+ LeftNet(2))!Green
       BmpOut(m*3+3,n)=BmpIn((m+ LeftNet(1))*PixMult+LES,n+RightNet(2))!Red
     enddo!i
   enddo!j
   endif !I1.BytesPerPixel=1

   if(PixMult.eq.3) then
   do   n= 1,I2.PixelHeight
     do m= 0,I2.PixelsPerRow-1
       BmpOut(m*3+1,n)=BmpIn((m+RightNet(1))*PixMult+1+RES,n+ LeftNet(2))!Blue
       BmpOut(m*3+2,n)=BmpIn((m+RightNet(1))*PixMult+2+RES,n+ LeftNet(2))!Green
       BmpOut(m*3+3,n)=BmpIn((m+ LeftNet(1))*PixMult+3+LES,n+RightNet(2))!Red
     enddo!i
   enddo!j
   endif !I1.BytesPerPixel=3

                             call Fdate20(DaTime);write(6,"(59x,a20)") DaTime
   write(6,*) 'Program RedGB3d.exe:'
   write(6,*) '  is about to write output file:',I2.BmpFileName
   write(6,*) '  which is',I2.BH.nWidth,'pixels by',I2.BH.nHeight,'pixels.'

   write(6,"(/15x,'Press 0 to exit now  -or-  Press 1 to proceed:',\)")
   read( 5,*) Icontinue
   if(Icontinue.ne.1) then
     pause 'program halting... press enter.';stop;
   endif!(Icontinue<>1)
   write(6,*);               call Fdate20(DaTime);write(6,"(59x,a20)") DaTime

  !Exporting the image to disk:
   open(unit=iUnitR,file=I2.BmpFileName,form='binary',action='write')
     write(  iUnitR)     I2.BH, BmpOut
   close(    iUnitR)
   deallocate(BmpIn )
   deallocate(Bmpout)
                             call Fdate20(DaTime);write(6,"(59x,a20)") DaTime

   pause'RedGB3D.exe image processing completed. Press enter.'
End Program RedGB3D !------------------------------------------------------

Subroutine ReadBmpImageHeader(I) !---------------------------------------------
!2015.09.21.1000cdt JMS- Reads a bitmap's BH header:
!                      - Traveler2/Athlon64/WinXPPro/APF9.0
!-----
 use RedGB3DDec,only:ImageRec,iUnitR,Exists
 implicit none
  type(ImageRec)::I  !=Ir(1) or =Ir(2)
  !----------                                                End of declarations

   !Checking whether or not the input file exits... a minimal precaution:
   write(6,*) 'Searching for ',I.BmpFileName
   inquire(file=I.BmpFileName,exist=Exists)
     if(.NOT. Exists)then;
       write(6,"(1x,'...not found.',a1\1x)") '\a'C
       pause '     RedGB3D will exit... press enter.' ; stop' Halt.' ;
     end if!(!Exists)
   write(6,*) '...found.  Opening...'

   !Read BmpFileIn's header record:
   open(unit=iUnitR,file=I.BmpFileName,form='binary',action='read')
     read(   iUnitR) I.BH
   close(    iUnitR)

   call PrintBmpImageHeader(I.BH)

   I.PixelsPerRow = I.BH.nWidth
   I.PixelHeight  = I.BH.nHeight

   !This program (version 0.5) converts a subset of all possible .bmp's:
   if((I.BH.nBitsPP.ne.8).and.(I.BH.nBitsPP.ne.24)) then
     pause 'Vsn 0.5 only supports 8bit grayscale & 24bit bit color .bmp files.'
     stop
   endif;

   !Preparing to import the .bmp image in one fell swoop:
   I.BytesPerPixel     =   I.BH.nBitsPP/8
   I.BytesPerRow       =   I.BH.nWidth*I.BytesPerPixel
   I.BytesPerRowPadded =  (I.BH.nSizeTot-54)/I.BH.nHeight
   I.PadBytes          =   I.BytesPerRowPadded   &
                         - I.BytesPerRow
   write(6,"(/'I',i1,'.PixelsPerRow     =',i8)") I.N, I.PixelsPerRow
!  write(6,"( 'I',i1,'.PixelsConvAdj    =',i8)") I.N, I.PixelsConvAdj
   write(6,"( 'I',i1,'.BytesPerPix      =',i8)") I.N, I.BytesPerPixel
   write(6,"( 'I',i1,'.BytesPerRow      =',i8)") I.N, I.BytesPerRow
   write(6,"( 'I',i1,'.PadBytes         =',i8)") I.N, I.PadBytes
   write(6,"( 'I',i1,'.BytesPerRowPadded=',i8)") I.N, I.BytesPerRowPadded
  return
End Subroutine ReadBmpImageHeader !------------------------------------------

Subroutine PrintBmpImageHeader(BH) !----------------------------------------
!2015.09.21.1145cdt JMS- This code prints the BH Header values:
!                      - Traveler2/Athlon64/WinXPPro/APF9.0
!-----
 use RedGB3DDec, only: BmpHdr
 implicit none
  type(BmpHdr)::BH
  !----------                                             End of declarations

   !Display the 16 header values:
   write(6,"('  Bitmap Header (BH) values:')")
   write(6,"('  BM=',a2  ,12x  ,' nSizeTot='  ,i10\ )") BH.BM      ,BH.nSizeTot
   write(6,"('  nReserv1=',i8  ,'   nReserv2=',i8   )") BH.nReserv1,BH.nReserv2
   write(6,"('  nOffBit =',i8  ,'   nSizeH  =',i8\  )") BH.nOffBit ,BH.nSizeH
   write(6,"('  nWidth  =',i8  ,'   nHeight =',i8   )") BH.nWidth  ,BH.nHeight
   write(6,"('  nPlanes =',i8  ,'   nBitsPP =',i8\  )") BH.nPlanes ,BH.nBitsPP
   write(6,"('  nCompres=',i8  ,'   nSizeC  =',i8   )") BH.nCompres,BH.nSizeC
   write(6,"('  XPpM    =',f8.2,'   YPpM    =',f8.2\)") BH.XPpM   ,BH.YPpM
   write(6,"('  nClrUsed=',i8  ,'   nClrImpo=',i8   )") BH.nClrUsed,BH.nClrImpo

 return
End Subroutine PrintBmpImageHeader !----------------------------------------

Subroutine Fdate20(DaTime) !------------------------------------------------
!2012.07.23.0040cdt JMS- Acquire the current date & Time:
!                      - Traveler2/Athlon64/WinXPPro/APF9.0
 implicit none
  character::DaTime*20
  integer*4::i,iDMY(3),iHMS(3)
!----------
   call iDate(iDMY)
   call iTime(iHMS)
   write(DaTime,79) (iDMY(i),i=3,1,-1),(iHMS(i),i=1,3),0
79 format(i4,'.',i2.2,'.',i2.2,'.',i2.2,i2.2'.',i2.2,' L')
   return
End Subroutine Fdate20 !----------------------------------------------------

```