

File created: 18-Oct-2023 21:50:02 {DSK}<home>frank<il>notecards>library>bitmapeditor.;2

changes to: (FNS EditBitmapBitmapFromFile EditBitmapFileRestore EditBitmapFileSave EditBitmapAutoSave
EditBitmapFetchPatterns EditBitmapStorePatterns)

previous date: 16-Nov-88 09:35:10 {DSK}<home>frank<il>notecards>library>bitmapeditor.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

(RPAQQ **BITMAPEDITORCOMS**

```
[( * * Basic window and editor functions)
(FNS EditBitmap EditBitmapBySections EditBitmapChooseSubitem EditBitmapClose EditBitmapConfirmDiscard
EditBitmapDisplayBitmap EditBitmapDoEditItem EditBitmapMakeDirty EditBitmapMakeMenu
EditBitmapMakeUndirty EditBitmapMenuSelectionFn EditBitmapPopUpMenu EditBitmapQuit
EditBitmapRepaintFn EditBitmapSetError EditBitmapSetQuit EditBitmapSetStop EditBitmapSetWindowProps
EditBitmapStop EditBitmapWaitForFinished)
(* * Editing functions)
(FNS EditBitmapAddBitmap EditBitmapAddBorder EditBitmapAddTexture EditBitmapBitmapEditRegion
EditBitmapBitmapFromFile EditBitmapBlock? EditBitmapChangeBitmapSize
EditBitmapChangeBitmapSizeMaybe EditBitmapCheckpointRestore EditBitmapCheckpointSave
EditBitmapClearAllButRegion EditBitmapClearBitmap EditBitmapClearRegion EditBitmapCopyCreate
EditBitmapCopyEdit EditBitmapCopyOperate EditBitmapDoInversion EditBitmapDoRotation
EditBitmapDrawArrowhead EditBitmapDrawArrowheadFilled EditBitmapDrawArrowheadOpen EditBitmapDrawBox
EditBitmapDrawCircle EditBitmapDrawCurve EditBitmapDrawCurveClosed EditBitmapDrawCurveOpen
EditBitmapDrawEllipse EditBitmapDrawGrid EditBitmapDrawGridRegion EditBitmapDrawGridWindow
EditBitmapDrawLines EditBitmapDrawPolygon EditBitmapDrawText EditBitmapExpandBitmap
EditBitmapFileRestore EditBitmapFileSave EditBitmapFillBox EditBitmapFillBoxSpecified
EditBitmapFillRegion EditBitmapFillRegionDefault EditBitmapFillRegionSpecified EditBitmapGetShade
EditBitmapGetSizeFactor EditBitmapInvertBitmap EditBitmapInvertColor EditBitmapInvertColorRegion
EditBitmapMakeExact EditBitmapMakeSmaller EditBitmapMask EditBitmapMoveRectangularRegion
EditBitmapPaintWithAirbrush EditBitmapPaintWithBitmap EditBitmapPaintWithBrush
EditBitmapPatternArray EditBitmapPixelEditRegion EditBitmapPixelEditWindow EditBitmapPlacePattern
EditBitmapResetBitmap EditBitmapRotateBitmap EditBitmapSaveBitmap EditBitmapSetPatternAttribute
EditBitmapSetPatternDone EditBitmapSetPatternMenu EditBitmapSetSide EditBitmapSetSideDone
EditBitmapSetSideMenu EditBitmapShiftBitmap EditBitmapShrinkBitmap EditBitmapTesselate
EditBitmapTesselateBitmap EditBitmapTesselateRegion EditBitmapTrimBitmap EditBitmapTrimSides
EditBitmapUndo)
(* * Averaging functions)
(FNS EditBitmapAverageBitInBitmap EditBitmapAveraging EditBitmapBitmapAverage EditBitmapGetAveragingInfo
EditBitmapShowAveragingFormat)
(* * Auto-Save functions)
(FNS EditBitmapAutoSave EditBitmapSetAutoSave EditBitmapSetAutoSaveFile EditBitmapSetAutoSaveInterval
EditBitmapShowAutoSave)
(* * Magnification functions)
(FNS EditBitmapMagnify EditBitmapMagnifyBitmap EditBitmapMagnifyChange EditBitmapMagnifyClose
EditBitmapMagnifyEdit EditBitmapMagnifyEditButtonEventFn EditBitmapMagnifyMoved
EditBitmapMagnifySelectButtonEventFn)
(* * Pixel follow funtions)
(FNS EditBitmapDrawOverPixels EditBitmapDrawOverPixelsRegion EditBitmapDrawOverPixelsSingle
EditBitmapErasePixelsRegion EditBitmapErasePixelsSingle EditBitmapFollowPixelsMatrix
EditBitmapFollowPixelsMatrixClose EditBitmapFollowPixelsMatrixDone
EditBitmapFollowPixelsMatrixSelect EditBitmapFollowPixelsRegion EditBitmapFollowPixelsSingle
EditBitmapRemovePixels EditBitmapSelectBrushOffset)
(* * Pattern functions)
(FNS EditBitmapAddPattern EditBitmapCopyEditPattern EditBitmapDeletePattern EditBitmapEditPattern
EditBitmapFetchPatterns EditBitmapFontStylesheet EditBitmapGetPattern EditBitmapMakePattern
EditBitmapPatternMenu EditBitmapShowPattern EditBitmapStorePatterns EditBitmapTrimPattern)
(* * Distort functions)
(FNS EditBitmapBitmapDistort EditBitmapDistort EditBitmapDistortBitmap EditBitmapDrawQuadrilateral
EditBitmapExpandQuadrilateral EditBitmapGetDeltas EditBitmapGetExtents EditBitmapGetNewVertices
EditBitmapGetQuadrilateral)
(* * Parameter display and setting functions)
(FNS EditBitmapConfirmDashing EditBitmapGetNewDashing EditBitmapOperationMenu EditBitmapSetAirBrushSize
EditBitmapSetAirBrushSpeed EditBitmapSetArrowhead EditBitmapSetBrushShape EditBitmapSetDashing
EditBitmapSetDrawBrushSize EditBitmapSetFont EditBitmapSetGrid EditBitmapSetOperation
EditBitmapSetOrthogonal EditBitmapSetPaintBrushSize EditBitmapSetRegionGrid EditBitmapSetShade
EditBitmapShowParameters)
(* * Interface to TEdit and SKETCH)
(FNS EditBitmapImageObjButtonEventInFn EditBitmapImageObjectCreate EditBitmapInsertBitmapIntoSketch
EditBitmapInsertBitmapIntoTedit EditBitmapSelectTargetWindow)
(* * Very basic input/output functions)
(FNS EditBitmapAirbrushOutline EditBitmapAirbrushPaint EditBitmapArrowhead
EditBitmapBitmapInvertDiagonal EditBitmapBitmapInvertHorizontal EditBitmapBitmapInvertVertical
EditBitmapBitmapRotate EditBitmapBitmapRotateArbitrary EditBitmapBitmapShift EditBitmapBorder
EditBitmapBoxInput EditBitmapCircleGetRadiusPoint EditBitmapCircleInput EditBitmapCircleShow
EditBitmapConstrainRegion EditBitmapCopyBitmap EditBitmapCreatePixelSelectionCursor
EditBitmapCursorFields EditBitmapCurveInput EditBitmapDistance EditBitmapDrawOverConnectedPixels
EditBitmapEditRegion EditBitmapEllipseGetMajorPoint EditBitmapEllipseGetMinorPoint
EditBitmapEllipseInput EditBitmapEllipseOrientation EditBitmapEllipseShowMajor
EditBitmapEllipseShowMinor EditBitmapFromScreen EditBitmapGetFileName EditBitmapGetGriddedValue
EditBitmapGetPoint EditBitmapGetPointAnywhere EditBitmapGetPointList EditBitmapGetPosition
EditBitmapGetScreenBitmap EditBitmapGetScreenRegion EditBitmapGetStringFromUser
EditBitmapGetWindowRegion EditBitmapInsideWindow EditBitmapInsureGriddedRegion
EditBitmapLocatePixelOn EditBitmapMakeGrid EditBitmapMakeMask EditBitmapMarkSpot EditBitmapMessage
```

```

EditBitmapMessageClose EditBitmapMoveRegion EditBitmapPaintWindow EditBitmapPaintWindowWithBitmap
EditBitmapPlaceBitmap EditBitmapRandomBitWithinBrush EditBitmapReadBitmap
EditBitmapRemoveConnectedPixels EditBitmapReshapeWindow EditBitmapRound EditBitmapScreenInput
EditBitmapShowPoint EditBitmapSmaller EditBitmapTrackBitmap EditBitmapTrim
EditBitmapVerticesFromRegion EditBitmapWireInput EditBitmapWireShowClosed EditBitmapWireShowOpen
EditBitmapWriteBitmap)
(* * Cursors)
(CURSORS EditBitmapCircleCenter EditBitmapCircleEdge EditBitmapEllipseCenter EditBitmapEllipseMajor
  EditBitmapEllipseMinor EditBitmapMagnifyCursor)
(* * Variables)
(VARS EditBitmapAirbrushTimerIntervals EditBitmapBlockTime EditBitmapDefaultAirBrushSize
  EditBitmapDefaultArrowHeight EditBitmapDefaultArrowWidth EditBitmapDefaultAutoSaveDeltaTime
  EditBitmapDefaultAveraging EditBitmapDefaultBrushSize EditBitmapDefaultFont
  EditBitmapDefaultListOfDashings EditBitmapDefaultListOfTextures EditBitmapDefaultPaintBrushSize
  EditBitmapMaxArrowSize EditBitmapMaxBrushSize EditBitmapMaxGridSize EditBitmapMaxPatternSize
  EditBitmapMenuFont EditBitmapMenuItems EditBitmapMenuPointer EditBitmapMessageFont
  EditBitmapMinArrowSize EditBitmapMinPatternSize EditBitmapMinRegionSize EditBitmapMinSize
  EditBitmapPixelSelectBoxSize EditBitmapSpotMarker)
(GLOBALVARS EditBitmapAirbrushTimerIntervals EditBitmapBlockTime EditBitmapCircleCenter
  EditBitmapCircleEdge EditBitmapDefaultAirBrushSize EditBitmapDefaultArrowHeight
  EditBitmapDefaultArrowWidth EditBitmapDefaultAutoSaveDeltaTime EditBitmapDefaultAveraging
  EditBitmapDefaultBrushSize EditBitmapDefaultFont EditBitmapDefaultListOfDashings
  EditBitmapDefaultListOfTextures EditBitmapDefaultPaintBrushSize EditBitmapEllipseCenter
  EditBitmapEllipseMajor EditBitmapEllipseMinor EditBitmapListOfDashings EditBitmapListOfTextures
  EditBitmapMagnifyCursor EditBitmapMaxArrowSize EditBitmapMaxBrushSize EditBitmapMaxGridSize
  EditBitmapMaxPatternSize EditBitmapMenuFont EditBitmapMenuItems EditBitmapMenuPointer
  EditBitmapMessageFont EditBitmapMinArrowSize EditBitmapMinPatternSize EditBitmapMinRegionSize
  EditBitmapMinSize EditBitmapPixelSelectBoxSize EditBitmapSpotMarker)
(* * Auxiliary files)
(FILE READNUM (SYSLOAD FROM LISPUSERS)
  FILLREGION)
(* * Initialization)
(P (SETQ EditBitmapListOfTextures (COPYALL EditBitmapDefaultListOfTextures))
  (SETQ EditBitmapListOfDashings (COPYALL EditBitmapDefaultListOfDashings))

```

```
(* * Basic window and editor functions)
```

```
(DEFINEQ
```

(EditBitmap

```

[LAMBDA (BitmapOrFile Window NoQuitItem? OtherMenuItems MaxWidth MaxHeight)
  (* Gaska "6-Oct-88 10:40")

```

```
(* * Implements a bitmap editor)
```

```

(LET* [[Bitmap (COND
  ((BITMAPP BitmapOrFile)
    (EditBitmapCopyBitmap BitmapOrFile))
  ((STRINGP BitmapOrFile)
    (EditBitmapBitmapFromFile BitmapOrFile))
  (BitmapOrFile (PRIN1 (CONCAT "Argument is neither a bitmap nor a string (file name)."
    (CHARACTER 13)))
    NIL)
  (Window (LET [(WindowImage (BITMAPCREATE (WINDOWPROP Window 'WIDTH)
    (WINDOWPROP Window 'HEIGHT)
    (BITBLT Window NIL NIL WindowImage)
    WindowImage))
    (T (EditBitmapFromScreen EditBitmapMinSize EditBitmapMinSize (OR MaxWidth SCREENWIDTH)
    (OR MaxHeight SCREENHEIGHT)
    (Window (AND Bitmap (OR Window (CREATEW (GETBOXREGION (WIDTHIFWINDOW (BITMAPWIDTH Bitmap)
    4)
    (HEIGHTIFWINDOW (BITMAPHEIGHT Bitmap)
    4)
    T 4))
    "Bitmap Editor"]
    (if Bitmap
      then (RESETLST
        (RESETSAVE (CURSOR WAITINGCURSOR))
        (EditBitmapMakeMenu Window (APPEND EditBitmapMenuItems OtherMenuItems)
          NoQuitItem?)
        (EditBitmapOperationMenu Window 'PAINT)
        (EditBitmapSetWindowProps Window Bitmap NoQuitItem?)
        (REDISPLAYW Window))
      (if (NULL NoQuitItem?)
        then (SELECTQ (EditBitmapWaitForFinished Window)
          (Quit (EditBitmapQuit Window))
          (Stop (EditBitmapClose Window)
            NIL)
          (Error (EditBitmapClose Window)
            (ERROR!))
          NIL)
        else Window)
      else (ERROR!)]

```

(EditBitmapBySections

```

[LAMBDA (Bitmap)
  (* Gaska "21-Sep-88 13:46")

```

(* * Edit a large bitmap by sections)

```
(LET* ((Position (CREATEPOSITION LASTMOUSEX LASTMOUSEY))
      (Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (SectionWidth (READNUM "Section Width" Position EditBitmapMessageFont (IMIN Width 100)
                             Width NIL T T))
      (SectionHeight (READNUM "Section Height" Position EditBitmapMessageFont (IMIN Height 100)
                              Height NIL T T))
      (Overlap (READNUM "Overlap" Position EditBitmapMessageFont 10 100 NIL T T))
      (NewBitmap (EditBitmapCopyBitmap Bitmap)))
[for Y0 from 0 to (SUB1 Height) by SectionHeight
 do (for X0 from 0 to (SUB1 Width) by SectionWidth
    do (LET* [(SectionBitmap (BITMAPCREATE (IMIN (ADD1 (IDIFFERENCE Width X0))
                                              (IPLUS SectionWidth Overlap))
      (IMIN (ADD1 (IDIFFERENCE Height Y0))
      (IPLUS SectionHeight Overlap))
      (BITBLT NewBitmap X0 Y0 SectionBitmap 0 0 (IPLUS SectionHeight Overlap)
      (IPLUS SectionHeight Overlap))
      (LET [(NewSection (NLSETQ (EditBitmap SectionBitmap)
      (if (LISTP NewSection)
          then (if (CAR NewSection)
                  then (BITBLT (CAR NewSection)
                               0 0 NewBitmap X0 Y0 (IPLUS SectionHeight Overlap)
                               (IPLUS SectionHeight Overlap)))
          else (ERROR!])

      NewBitmap])])
```

(EditBitmapChooseSubitem

[LAMBDA (Window)

(* Gaska "15-Sep-88 15:17")

(* * Inform the user that it must choose a menu subitem)

```
(EditBitmapMessageClose Window)
(EditBitmapMessage Window "Choose a menu subitem")]
```

(EditBitmapClose

[LAMBDA (Window)

(* Gaska "20-Sep-88 14:46")

(* * Close down the bitmap editor)

```
(DETACHALLWINDOWS Window)
(WINDOWPROP Window 'Bitmap NIL)
(WINDOWPROP Window 'OriginalBitmap NIL)
(WINDOWPROP Window 'CopyOfBitmap NIL)
(WINDOWPROP Window 'SavedBitmap NIL)
(WINDOWPROP Window 'CheckPoint NIL)
(WINDOWPROP Window 'Patterns NIL)
(WINDOWADDPROP Window 'AveragingList NIL)
(WINDOWPROP Window 'OperationsMenu NIL)
(if (WINDOWPROP Window 'MessageWindow)
    then (CLOSEW (WINDOWPROP Window 'MessageWindow NIL)))
(WINDOWPROP Window 'Monitor NIL)
(WINDOWPROP Window 'CLOSEFN NIL)
(CLOSEW Window)]
```

(EditBitmapConfirmDiscard

[LAMBDA (Window)

(* Gaska " 6-Sep-88 08:47")

(* * Confirm discarding of changes)

```
(RESETLST
 (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
 (if (WINDOWPROP Window 'Dirty)
     then (LET NIL (EditBitmapMessage Window '("Bitmap has been changed" "Do you really want to discard
changes?" "Click left to confirm, right to abort."))
         (MOUSECONFIRM "" ""))
     else T)))
```

(EditBitmapDisplayBitmap

[LAMBDA (Window)

(* Gaska " 7-Sep-88 10:40")

(* * Display the changed bitmap)

```
(REDISPLAYW Window)
T])
```

(EditBitmapDoEditItem

[LAMBDA (Function Window)

(* Gaska "19-Sep-88 16:16")

(* * Perform the selected edit function)

```

(EditBitmapMessageClose Window)
(WINDOWPROP Window 'LastDirtyState (WINDOWPROP Window 'Dirty))
(EditBitmapSaveBitmap Window)
(if (AND Function (APPLY* Function Window))
    then (EditBitmapMakeDirty Window)
        (EditBitmapAutoSave Window))

```

(EditBitmapMakeDirty

```
[LAMBDA (Window) (* Gaska "18-Sep-88 11:42")
```

```
    (* * Mark the bitmap as changed)
```

```
(WINDOWPROP Window 'Dirty T])
```

(EditBitmapMakeMenu

```
[LAMBDA (Window Items NoQuitItem?) (* Gaska "22-Sep-88 08:25")
```

```
    (* * Create and attach the edit menu)
```

```

(ATTACHMENU (create MENU
    TITLE _ "Bitmap Edit Menu"
    ITEMS _ [APPEND Items (if (NULL NoQuitItem?)
        then '("Quit" (EditBitmapSetQuit)
            "Terminate editing and return a new bitmap"
            (SUBITEMS ("Quit, Return New Bitmap" (
                EditBitmapSetQuit
            )
                "Terminate editing and return a
                new bitmap")
            ("Quit, Return NIL" (EditBitmapSetStop)
                "Terminate editing and return NIL")
            ("Abort" (EditBitmapSetError)
                "Error out of the bitmap editor"]
        )
    MENUFONT _ EditBitmapMenuFont
    CENTERFLG _ T
    WHENSELECTEDFN _ 'EditBitmapMenuSelectionFn)
    Window
    'RIGHT
    'TOP])

```

(EditBitmapMakeUndirty

```
[LAMBDA (Window) (* Gaska "18-Sep-88 11:43")
```

```
    (* * Mark the bitmap as unchanged)
```

```
(WINDOWPROP Window 'Dirty NIL])
```

(EditBitmapMenuSelectionFn

```
[LAMBDA (Item Menu Button) (* Gaska "21-Sep-88 09:46")
```

```

    (if (OBTAIN.MONITORLOCK (WINDOWPROP (MAINWINDOW (WFROMMENU Menu))
        'Monitor)
        T)
    then (SHADEITEM Item Menu GRAYSHADE)
    [LET [(Window (MAINWINDOW (WFROMMENU Menu))
        (if (LISTP (CADR Item))
            then (EditBitmapMessageClose Window)
                (APPLY* (CAADR Item)
                    Window)
            else (EditBitmapDoEditItem (CADR Item)
                Window))
        (RELEASE.MONITORLOCK (WINDOWPROP Window 'Monitor)]
        (SHADEITEM Item Menu WHITESHADE])

```

(EditBitmapPopUpMenu

```
[LAMBDA (Items Title) (* Gaska "19-Sep-88 13:37")
```

```
    (* * Create a pop-up menu and return the response)
```

```

(MENU (create MENU
    ITEMS _ Items
    TITLE _ Title
    CENTERFLG _ T
    MENUFONT _ EditBitmapMenuFont
    MENUBORDERSIZE _ 1])

```

(EditBitmapQuit

```
[LAMBDA (Window) (* Gaska "26-Sep-88 08:13")
```

```
    (* * Close the bitmap editor and return the new bitmap)
```

```
(LET* ((Width (WINDOWPROP Window 'WIDTH))
      (Height (WINDOWPROP Window 'HEIGHT))
      (NewBitmap (BITMAPCREATE Width Height)))
  (BITBLT Window 0 0 NewBitmap 0 0 Width Height)
  (EditBitmapClose Window)
  NewBitmap])
```

(EditBitmapRepaintFn

[LAMBDA (Window)

(* Gaska "26-Sep-88 08:38")

(* * RepaintFn for the bitmap editor)

```
(LET* [(Bitmap (WINDOWPROP Window 'Bitmap)
  (BITBLT Bitmap 0 0 Window 0 0 (BITMAPWIDTH Bitmap)
    (BITMAPHEIGHT Bitmap)])
```

(EditBitmapSetError

[LAMBDA (Window)

(* Gaska "21-Sep-88 13:34")

(* * Set the finished flag to Error)

```
(WINDOWPROP Window 'Finished (AND (EditBitmapConfirmDiscard Window)
  'Error])
```

(EditBitmapSetQuit

[LAMBDA (Window)

(* Gaska "23-Aug-88 10:42")

(* * Set the finished flag to Quit)

```
(WINDOWPROP Window 'Finished 'Quit)
NIL])
```

(EditBitmapSetStop

[LAMBDA (Window)

(* Gaska "6-Sep-88 08:47")

(* * Set the finished flag to Stop)

```
(WINDOWPROP Window 'Finished (AND (EditBitmapConfirmDiscard Window)
  'Stop])
```

(EditBitmapSetWindowProps

[LAMBDA (Window Bitmap NoQuitItem?)

(* Gaska "23-Sep-88 11:07")

(* * Set window properties for the bitmap editor)

(* * Standard WINDOWPROPS)

```
(if (NULL NoQuitItem?)
  then (WINDOWADDPROP Window 'CLOSEFN 'DON'T))
(WINDOWPROP Window 'REPAINTFN 'EditBitmapRepaintFn)
(WINDOWPROP Window 'RESHAPEFN 'DON'T)
```

(* * Bitmaps and related)

```
(WINDOWPROP Window 'Bitmap Bitmap)
(WINDOWPROP Window 'OriginalBitmap (EditBitmapCopyBitmap Bitmap))
(WINDOWPROP Window 'SavedBitmap (BITMAPCREATE SCREENWIDTH SCREENHEIGHT))
(WINDOWPROP Window 'SavedBitmapWidth (BITMAPWIDTH Bitmap))
(WINDOWPROP Window 'SavedBitmapHeight (BITMAPHEIGHT Bitmap))
(WINDOWPROP Window 'CheckPoint (EditBitmapCopyBitmap Bitmap))
(WINDOWPROP Window 'Dirty NIL)
```

(* * Parameters)

```
(WINDOWPROP Window 'BrushShape 'ROUND)
(WINDOWPROP Window 'BrushSize EditBitmapDefaultBrushSize)
(WINDOWPROP Window 'PaintBrushSize EditBitmapDefaultPaintBrushSize)
(WINDOWPROP Window 'AirBrushSize EditBitmapDefaultAirBrushSize)
(WINDOWPROP Window 'AirBrushSpeed 'FAST)
(WINDOWPROP Window 'Dashing NIL)
(WINDOWPROP Window 'Font EditBitmapDefaultFont)
(WINDOWPROP Window 'ArrowWidth EditBitmapDefaultArrowWidth)
(WINDOWPROP Window 'ArrowHeight EditBitmapDefaultArrowHeight)
(WINDOWPROP Window 'Shade BLACKSHADE)
(WINDOWPROP Window 'RegionGrid NIL)
(WINDOWPROP Window 'Grid NIL)
(WINDOWPROP Window 'Orthogonal NIL)
(WINDOWPROP Window 'Operation 'PAINT)
```

(* * Miscellaneous)

```
(WINDOWPROP Window 'Monitor (CREATE.MONITORLOCK 'EditBitmap))
```

```
(WINDOWPROP Window 'Patterns NIL)
(WINDOWPROP Window 'AveragingList NIL)
(WINDOWPROP Window 'AutoSave NIL)
(WINDOWPROP Window 'AutoSaveFileName NIL)
(WINDOWPROP Window 'AutoSaveDeltaTime (ITIMES EditBitmapDefaultAutoSaveDeltaTime 60))
(WINDOWPROP Window 'AutoSaveLastSave (IDATE])
```

(EditBitmapStop

[LAMBDA (Window)

(* Gaska "22-Aug-88 12:51")

(* * Terminate the editing process without saving any changes)

(EditBitmapClose Window])

(EditBitmapWaitForFinished

[LAMBDA (Window)

(* Gaska "18-Sep-88 12:40")

(* * Loop until the bitmap editor is finished)

```
(SPAWN.MOUSE)
(while (NULL (WINDOWPROP Window 'Finished)) do (DISMISS 500))
(WINDOWPROP Window 'Finished])
```

)

(* * Editing functions)

(DEFINEQ

(EditBitmapAddBitmap

[LAMBDA (Window)

(* Gaska "5-Oct-88 10:36")

(* * Add a bitmap from the screen)

```
(LET* ((BitmapAndPosition (EditBitmapScreenInput Window EditBitmapMinSize EditBitmapMinSize SCREENWIDTH
                                SCREENHEIGHT)))
  (if BitmapAndPosition
    then (BITBLT (CAR BitmapAndPosition)
                 NIL NIL (WINDOWPROP Window 'Bitmap)
                 (CAADR BitmapAndPosition)
                 (CDADR BitmapAndPosition)
                 NIL NIL 'INPUT (WINDOWPROP Window 'Operation))
    (EditBitmapDisplayBitmap Window]))
```

(EditBitmapAddBorder

[LAMBDA (Window)

(* Gaska "2-Sep-88 10:30")

(* * Add a border to the bitmap.)

```
(LET ((Bits (READNUM "Border Width" NIL EditBitmapMenuFont 1 200 NIL T)))
  (if Bits
    then (EditBitmapChangeBitmapSize Window (EditBitmapBorder (WINDOWPROP Window 'Bitmap)
                                                                Bits
                                                                (OR (EditBitmapGetShade)
                                                                    (WINDOWPROP Window 'Shade]))
```

(EditBitmapAddTexture

[LAMBDA (Window)

(* Gaska "20-Sep-88 08:36")

(* * Fill the bitmap with a texture)

```
(LET ((Shade (EditBitmapGetShade)))
  (if Shade
    then (BLTSHADE Shade (WINDOWPROP Window 'Bitmap)
                        NIL NIL NIL NIL (WINDOWPROP Window 'Operation))
    (EditBitmapDisplayBitmap Window]))
```

(EditBitmapBitmapEditRegion

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:13")

(* * Edit a region)

```
(if (EditBitmapEditRegion Window (WINDOWPROP Window 'Bitmap)
                             'EditBitmap
                             (WINDOWPROP Window 'RegionGrid)
                             (WINDOWPROP Window 'Grid)
                             EditBitmapMinRegionSize EditBitmapMinRegionSize)
  then (EditBitmapDisplayBitmap Window]))
```

(EditBitmapBitmapFromFile

```
[LAMBDA (File) (* Gaska "6-Oct-88 10:34")
```

```
(* * Retrieve a bitmap from a file)
```

```
(if (AND File (FINDFILE File))
  then (LET [(Stream (OPENSTREAM File 'INPUT)
                (RESETLST
                 (RESETSAVE (CURSOR WAITINGCURSOR))
                 (RESETSAVE NIL (LIST 'CLOSEF Stream))
                 (EditBitmapReadBitmap Stream)))
            else (PRIN1 (CONCAT "Can't find file: " File (CHARACTER 13)))
              (ERROR!))
```

(EditBitmapBlock?

```
[LAMBDA NIL (* Gaska "26-Oct-88 10:06")
```

```
(* * Determine if the user wants to share the cpu with a long running process)
```

```
(SELECTQ (EditBitmapPopUpMenu ' (Abort Yes No)
          "Monopolize CPU?")
  (Abort (ERROR!))
  (Yes NIL)
  (No EditBitmapBlockTime)
  NIL])
```

(EditBitmapChangeBitmapSize

```
[LAMBDA (Window NewBitmap) (* Gaska "8-Sep-88 13:10")
```

```
(* * Change the size of the bitmap)
```

```
(if NewBitmap
  then (RESETLST
    [for OtherWindow in (REVERSE (ATTACHEDWINDOWS Window))
      do (RESETSAVE (DETACHWINDOW OtherWindow)
        (LIST 'ATTACHWINDOW OtherWindow Window (CAR (WINDOWPROP OtherWindow
          'WHEREATTACHED))
          (CDR (WINDOWPROP OtherWindow 'WHEREATTACHED))
          (WINDOWPROP Window 'Bitmap NewBitmap)
          [LET* [(WindowRegion (WINDOWPROP Window 'REGION)
            (EditBitmapReshapeWindow Window (CREATEREGION (fetch LEFT of WindowRegion)
              (fetch BOTTOM of WindowRegion)
              (WIDTHIFWINDOW (BITMAPWIDTH NewBitmap)
                (WINDOWPROP Window 'BORDER))
              (HEIGHTIFWINDOW (BITMAPHEIGHT NewBitmap)
                (WINDOWPROP Window 'TITLE)
                (WINDOWPROP Window 'BORDER]))])
```

(EditBitmapChangeBitmapSizeMaybe

```
[LAMBDA (Window NewBitmap) (* Gaska "7-Sep-88 10:42")
```

```
(* * Change the size of the bitmap if necessary)
```

```
(if (OR (NEQ (BITMAPWIDTH (WINDOWPROP Window 'Bitmap))
  (BITMAPWIDTH NewBitmap))
  (NEQ (BITMAPHEIGHT (WINDOWPROP Window 'Bitmap))
  (BITMAPHEIGHT NewBitmap)))
  then (EditBitmapChangeBitmapSize Window NewBitmap)
  else (BITBLT NewBitmap NIL NIL (WINDOWPROP Window 'Bitmap))
    (EditBitmapDisplayBitmap Window])
```

(EditBitmapCheckpointRestore

```
[LAMBDA (Window) (* Gaska "3-Oct-88 08:32")
```

```
(* * Restore the bitmap to the last checkpoint state)
```

```
(if (EditBitmapConfirmDiscard Window)
  then (EditBitmapChangeBitmapSizeMaybe Window (WINDOWPROP Window 'CheckPoint]))
```

(EditBitmapCheckpointSave

```
[LAMBDA (Window) (* Gaska "16-Sep-88 15:24")
```

```
(* * Save the current state of the bitmap)
```

```
[WINDOWPROP Window 'CheckPoint (EditBitmapCopyBitmap (WINDOWPROP Window 'Bitmap)
  (EditBitmapMessage Window "Bitmap has been saved."))
```

(EditBitmapClearAllButRegion

```
[LAMBDA (Window) (* Gaska "23-Sep-88 11:11")
```

```
(* * Clear the entire bitmap except for a specified region)
```

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Select the region to be saved.")
  [LET* ((Region (EditBitmapGetWindowRegion Window (WINDOWPROP Window 'RegionGrid)
    EditBitmapMinRegionSize EditBitmapMinRegionSize)))
    (if Region
      then (LET [(TempBitmap (BITMAPCREATE (BITMAPWIDTH (WINDOWPROP Window 'Bitmap))
        (BITMAPHEIGHT (WINDOWPROP Window 'Bitmap)
        (BLTSHADE BLACKSHADE TempBitmap NIL NIL NIL NIL 'REPLACE)
        (BLTSHADE WHITESHADE TempBitmap (fetch LEFT of Region)
        (fetch BOTTOM of Region)
        (fetch WIDTH of Region)
        (fetch HEIGHT of Region)
        'REPLACE)
        (BITBLT TempBitmap NIL NIL (WINDOWPROP Window 'Bitmap)
        NIL NIL NIL NIL 'INPUT 'ERASE)
        (EditBitmapDisplayBitmap Window])))]

```

(EditBitmapClearBitmap

```
[LAMBDA (Window) (* Gaska "7-Sep-88 10:41")
```

```
(* * Clear the bitmap to all-white.)
```

```

(BLTSHADE WHITESHADE (WINDOWPROP Window 'Bitmap)
  0 0 (BITMAPWIDTH (WINDOWPROP Window 'Bitmap))
  (BITMAPHEIGHT (WINDOWPROP Window 'Bitmap))
  'REPLACE)
(EditBitmapDisplayBitmap Window])

```

(EditBitmapClearRegion

```
[LAMBDA (Window) (* Gaska "23-Sep-88 11:08")
```

```
(* * Clear a region)
```

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Select the region to be cleared.")
  [LET* ((Region (EditBitmapGetWindowRegion Window (WINDOWPROP Window 'RegionGrid)
    EditBitmapMinRegionSize EditBitmapMinRegionSize)))
    (if Region
      then (BLTSHADE WHITESHADE (WINDOWPROP Window 'Bitmap)
        (fetch LEFT of Region)
        (fetch BOTTOM of Region)
        (fetch WIDTH of Region)
        (fetch HEIGHT of Region)
        'REPLACE)
        (EditBitmapDisplayBitmap Window)))))]

```

(EditBitmapCopyCreate

```
[LAMBDA (Window) (* Gaska "20-Sep-88 09:21")
```

```
(* * Create a copy of the bitmap)
```

```

[WINDOWPROP Window 'CopyOfBitmap (EditBitmapCopyBitmap (WINDOWPROP Window 'Bitmap)
  (EditBitmapMessage Window "Copy has been created.")
  NIL])

```

(EditBitmapCopyEdit

```
[LAMBDA (Window) (* Gaska "19-Sep-88 16:02")
```

```
(* * Edit the copy of the bitmap)
```

```

[if (NULL (WINDOWPROP Window 'CopyOfBitmap))
  then (WINDOWPROP Window 'CopyOfBitmap (EditBitmapCopyBitmap (WINDOWPROP Window 'Bitmap)
  (LET [(NewBitmap (EditBitmap (WINDOWPROP Window 'CopyOfBitmap)
    (if NewBitmap
      then (WINDOWPROP Window 'CopyOfBitmap NewBitmap)))
    NIL])

```

(EditBitmapCopyOperate

```
[LAMBDA (Window) (* Gaska "19-Sep-88 16:17")
```

```
(* * Apply the current operation on the bitmap and the copy)
```

```

(if (WINDOWPROP Window 'CopyOfBitmap)
  then (BITBLT (WINDOWPROP Window 'CopyOfBitmap)
    0 0 (WINDOWPROP Window 'Bitmap)
    0 0 NIL NIL 'INPUT (WINDOWPROP Window 'Operation))
    (EditBitmapDisplayBitmap Window)
  else (EditBitmapMessage Window "No copy has been created.")]

```


(EditBitmapDoInversion

[LAMBDA (Window Function)

(* Gaska "26-Oct-88 10:07")

(* * Perform a bitmap inversion)

```

(LET* ((Block? (EditBitmapBlock?)))
  (EditBitmapChangeBitmapSizeMaybe Window (RESETLST
    (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
    (RESETSAVE (CURSOR (if Block?
      then T
      else WAITINGCURSOR))))
    (if Block?
      then (SPAWN.MOUSE))
    (EditBitmapMessage Window "This may take a LONG time!")
    (APPLY* Function (WINDOWPROP Window 'Bitmap)
      Block?))))

```

(EditBitmapDoRotation

[LAMBDA (Window Function Arg)

(* Gaska "26-Oct-88 10:08")

(* * Perform a bitmap rotation)

```

(LET* ((Block? (EditBitmapBlock?)))
  (EditBitmapChangeBitmapSizeMaybe Window (RESETLST
    (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
    (RESETSAVE (CURSOR (if Block?
      then T
      else WAITINGCURSOR))))
    (if Block?
      then (SPAWN.MOUSE))
    (EditBitmapMessage Window "This may take a LONG time!")
    (APPLY* Function (WINDOWPROP Window 'Bitmap)
      Arg Block?))))

```

(EditBitmapDrawArrowhead

[LAMBDA (Window Filled?)

(* Gaska "23-Sep-88 09:52")

(* * Draw a line with an arrowhead on the bitmap.)

```

(LET* [[Vertices (EditBitmapWireInput Window (WINDOWPROP Window 'Grid)
  (WINDOWPROP Window 'Orthogonal)
  [Stream (DSPCREATE (WINDOWPROP Window 'Bitmap)
    (EndOfLine (AND Vertices (LET* ([NextToLast (CAR (NTH Vertices (SUB1 (LENGTH Vertices)
      (XFirst (CAR NextToLast))
      (YFirst (CDR NextToLast))
      (XLast (CAAR (LAST Vertices)))
      (YLast (CDAR (LAST Vertices)))
      (DeltaX (DIFFERENCE XLast XFirst))
      (DeltaY (DIFFERENCE YLast YFirst)))
      (EditBitmapArrowhead Stream XLast YLast
        (if (ZEROP DeltaX)
          then (if (MINUSP DeltaY)
            then 270.0
            else 90.0)
          else (ARCTAN2 DeltaY DeltaX))
        (WINDOWPROP Window 'ArrowWidth)
        (WINDOWPROP Window 'ArrowHeight)
        (WINDOWPROP Window 'BrushSize)
        (WINDOWPROP Window 'Operation)
        (if Filled?
          then (WINDOWPROP Window 'Shade)]
      Vertices
    then (RPLACA (LAST Vertices)
      EndOfLine)
    for Position in Vertices first (DSPXPOSITION (CAAR Vertices)
      Stream)
    (DSPYPOSITION (CDAR Vertices)
      Stream)
    do (DRAWTO (CAR Position)
      (CDR Position)
      (WINDOWPROP Window 'BrushSize)
      (WINDOWPROP Window 'Operation)
      Stream
      (WINDOWPROP Window 'Shade)
      (WINDOWPROP Window 'Dashing]
      (EditBitmapDisplayBitmap Window)]

```

(EditBitmapDrawArrowheadFilled

[LAMBDA (Window)

(* Gaska "23-Aug-88 13:25")

(* * Draw an filled arrow at the end of a series of points)

```

(EditBitmapDrawArrowhead Window T])

```

(EditBitmapDrawArrowheadOpen

[LAMBDA (Window)

(* Gaska "23-Aug-88 13:25")

(* * Draw an unfilled arrow at the end of a series of points)

(EditBitmapDrawArrowhead Window))

(EditBitmapDrawBox

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:14")

(* * Draw a rectangle on the bitmap.)

```
(LET* [(Vertices (EditBitmapBoxInput Window (WINDOWPROP Window 'RegionGrid)
                                         EditBitmapMinRegionSize EditBitmapMinRegionSize))
      (Stream (DSPCREATE (WINDOWPROP Window 'Bitmap)
                          (if Vertices
                              then [for Position in Vertices first (DSPXPOSITION (CAAR Vertices)
                                          Stream)
                                   (DSPYPOSITION (CDAR Vertices)
                                          Stream)
                              do (DRAWTO (CAR Position)
                                         (CDR Position)
                                         (WINDOWPROP Window 'BrushSize)
                                         (WINDOWPROP Window 'Operation)
                                         Stream
                                         (WINDOWPROP Window 'Shade)
                                         (WINDOWPROP Window 'Dashing]
                                   (EditBitmapDisplayBitmap Window))
      ])
```

(EditBitmapDrawCircle

[LAMBDA (Window)

(* Gaska "7-Sep-88 10:33")

(* * Draw a Circle on the bitmap)

```
(LET* [(Circle (EditBitmapCircleInput Window (WINDOWPROP Window 'Grid)
      (if Circle
          then (LET [(Stream (DSPCREATE (WINDOWPROP Window 'Bitmap)
```

(* * Curve drawing functions do not accept a operation argument, so we change the operation in the stream.
Doesn't work correctly for all types of operations)

```
(DSPOPERATION (WINDOWPROP Window 'Operation)
  Stream)
(DRAWCIRCLE (CAAR Circle)
  (CDAR Circle)
  (EditBitmapDistance (CAR Circle)
    (CADR Circle))
  (LIST (WINDOWPROP Window 'BrushShape)
    (WINDOWPROP Window 'BrushSize))
  (WINDOWPROP Window 'Dashing)
  Stream)
(EditBitmapDisplayBitmap Window))
```

(EditBitmapDrawCurve

[LAMBDA (Window Closed?)

(* Gaska "7-Sep-88 10:51")

(* * Draw a curve on the bitmap)

```
(LET ((Points (EditBitmapCurveInput Window (WINDOWPROP Window 'Grid)
      T)))
```

```
(if Points
    then (LET [(Stream (DSPCREATE (WINDOWPROP Window 'Bitmap)
```

(* * Curve drawing functions do not accept a operation argument, so we change the operation in the stream.
Doesn't work correctly for all types of operations)

```
(DSPOPERATION (WINDOWPROP Window 'Operation)
  Stream)
(DRAWCURVE Points Closed? (LIST (WINDOWPROP Window 'BrushShape)
                                (WINDOWPROP Window 'BrushSize))
  (WINDOWPROP Window 'Dashing)
  Stream)
(EditBitmapDisplayBitmap Window))
```

(EditBitmapDrawCurveClosed

[LAMBDA (Window)

(* Gaska "22-Aug-88 08:03")

(* * Draw an closed curve on the bitmap)

(EditBitmapDrawCurve Window T))

(EditBitmapDrawCurveOpen

[LAMBDA (Window)

(* Gaska "22-Aug-88 08:03")

(* * Draw an open curve on the bitmap)

(EditBitmapDrawCurve Window)]

(EditBitmapDrawEllipse

[LAMBDA (Window)

(* Gaska " 7-Sep-88 11:23")

(* * Draw an ellipse on the bitmap)

```
(LET* ([Ellipse (EditBitmapEllipseInput Window (WINDOWPROP Window 'Grid]
  (Pos1 (CAR Ellipse))
  (Pos2 (CADDR Ellipse))
  (Pos3 (CADR Ellipse)))
  (if Ellipse
    then (LET [(Stream (DSPCREATE (WINDOWPROP Window 'Bitmap]
    (DSPOPERATION (WINDOWPROP Window 'Operation)
      Stream)
    (DRAWELLIPSE (CAR Pos1)
      (CDR Pos1)
      (EditBitmapDistance Pos1 Pos2)
      (EditBitmapDistance Pos1 Pos3)
      (EditBitmapEllipseOrientation Pos1 Pos3)
      (LIST (WINDOWPROP Window 'BrushShape)
        (WINDOWPROP Window 'BrushSize))
      (WINDOWPROP Window 'Dashing)
      Stream)
    (EditBitmapDisplayBitmap Window]))
```

(* * Curve drawing functions do not accept a operation argument, so we change the operation in the stream.
Doesn't work correctly for all types of operations)

```
(DSPOPERATION (WINDOWPROP Window 'Operation)
  Stream)
(DRAWELLIPSE (CAR Pos1)
  (CDR Pos1)
  (EditBitmapDistance Pos1 Pos2)
  (EditBitmapDistance Pos1 Pos3)
  (EditBitmapEllipseOrientation Pos1 Pos3)
  (LIST (WINDOWPROP Window 'BrushShape)
    (WINDOWPROP Window 'BrushSize))
  (WINDOWPROP Window 'Dashing)
  Stream)
(EditBitmapDisplayBitmap Window))
```

(EditBitmapDrawGrid

[LAMBDA (Window Region)

(* Gaska "12-Oct-88 16:35")

(* * Draw a grid)

```
(LET* [(Position (CREATEPOSITION LASTMOUSEX LASTMOUSEY))
  (Width (READNUM "Grid Width" Position EditBitmapMenuFont 5 50 NIL T))
  (Height (AND Width (READNUM "Grid Height" Position EditBitmapMenuFont 5 50 NIL T]
  (if Height
    then (EditBitmapMakeGrid (WINDOWPROP Window 'Bitmap)
      Region Width Height (WINDOWPROP Window 'BrushSize)
      (WINDOWPROP Window 'Operation)
      (WINDOWPROP Window 'Dashing)
      (EditBitmapPopupMenu ' ((Yes T)
        (No NIL))
        "Clip to Integer Number?"))
    (EditBitmapDisplayBitmap Window))
```

(EditBitmapDrawGridRegion

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:09")

(* * Draw a grid in a region of the window)

```
(LET [(Region (RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Select the region to be gridded")
  (EditBitmapGetWindowRegion Window (WINDOWPROP Window 'RegionGrid)
    EditBitmapMinRegionSize EditBitmapMinRegionSize))]
  (if Region
    then (EditBitmapDrawGrid Window Region))
```

(EditBitmapDrawGridWindow

[LAMBDA (Window)

(* Gaska " 7-Sep-88 11:29")

(* * Draw a grid on the entire bitmap)

```
(EditBitmapDrawGrid Window (CREATEREGION 0 0 (WINDOWPROP Window 'WIDTH)
  (WINDOWPROP Window 'HEIGHT]))
```

(EditBitmapDrawLines

[LAMBDA (Window)

(* Gaska " 7-Sep-88 10:57")

(* * Draw a line on the bitmap)

```
(LET [(Points (EditBitmapWireInput Window (WINDOWPROP Window 'Grid)
  (WINDOWPROP Window 'Orthogonal]
  (if Points
```

```

then (LET [(Stream (DSPCREATE (WINDOWPROP Window 'Bitmap)
(MOVETO (CAAR Points)
(CDAR Points)
Stream)
[for Point in (CDR Points) do (DRAWTO (CAR Point)
(CDR Point)
(WINDOWPROP Window 'BrushSize)
(WINDOWPROP Window 'Operation)
Stream
(WINDOWPROP Window 'Shade)
(WINDOWPROP Window 'Dashing]
(EditBitmapDisplayBitmap Window)])

```

(EditBitmapDrawPolygon

[LAMBDA (Window)

(* Gaska "7-Sep-88 10:58")

(* * Draw a polygon on the bitmap)

```

(LET ((Points (EditBitmapWireInput Window (WINDOWPROP Window 'Grid)
NIL T)))
(if Points
then (LET [(Stream (DSPCREATE (WINDOWPROP Window 'Bitmap)
(MOVETO (CAAR Points)
(CDAR Points)
Stream)
[for Point in (APPEND (CDR Points)
(LIST (CAR Points)))
do (DRAWTO (CAR Point)
(CDR Point)
(WINDOWPROP Window 'BrushSize)
(WINDOWPROP Window 'Operation)
Stream
(WINDOWPROP Window 'Shade)
(WINDOWPROP Window 'Dashing]
(EditBitmapDisplayBitmap Window)])

```

(EditBitmapDrawText

[LAMBDA (Window)

(* Gaska "12-Oct-88 09:30")

(* * Draw text on the bitmap)

```

(RESETLST
(RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
(DSPFONT (WINDOWPROP Window 'Font)
(WINDOWPROP Window 'DSP))
(LET* [[Text (RESETLST
(RESETSAVE NIL (LIST 'REMOVEPROMPTWINDOW Window))
(PROMPTFORWARD "Enter Text: " NIL NIL (GETPROMPTWINDOW Window 2 EditBitmapMessageFont)
NIL
'TTY))]]
(Bitmap (AND Text (LET* ([Region (STRINGREGION Text (WINDOWPROP Window 'DSP]
(Bitmap (BITMAPCREATE (fetch WIDTH of Region)
(fetch HEIGHT of Region)))
(Stream (DSPCREATE Bitmap)))
(DSPFONT (WINDOWPROP Window 'Font)
Stream)
(CENTERPRINTINREGION Text (CREATEREGION 0 0 (fetch WIDTH of Region)
(fetch HEIGHT of Region))
Stream)
Bitmap)))
(Position (AND Bitmap (EditBitmapPlaceBitmap Window Bitmap "Place the text" (WINDOWPROP
Window
'Grid]
(if Position
then (BITBLT Bitmap NIL NIL (WINDOWPROP Window 'Bitmap)
(CAR Position)
(CDR Position)
NIL NIL 'INPUT (WINDOWPROP Window 'Operation))
(EditBitmapDisplayBitmap Window))))))

```

(EditBitmapExpandBitmap

[LAMBDA (Window)

(* Gaska "7-Sep-88 11:00")

(* * Expand the bitmap)

```

(LET* ((Factors (EditBitmapGetSizeFactor "Expand")))
(if Factors
then (RESETLST
(RESETSAVE (CURSOR WAITINGCURSOR))
(EditBitmapChangeBitmapSize Window (EXPANDBITMAP (WINDOWPROP Window 'Bitmap)
(CAR Factors)
(CDR Factors)))))

```

(* Gaska "23-Sep-88 09:41")

(* Gaska "28-Oct-88 13:00")

(* Gaska "23-Sep-88 11:09")

(* Gaska "22-Aug-88 08:36")

(* Gaska " 8-Sep-88 08:04")

(* * Fill arbitrary shaped regions in the bitmap)

(EditBitmapFillRegionDefault

```
[LAMBDA (Window) (* Gaska "8-Sep-88 08:04")

(* * Fill arbitrary shaped regions in the bitmap with the default shade)

(EditBitmapFillRegion Window (WINDOWPROP Window 'Shade))
```

(EditBitmapFillRegionSpecified

```
[LAMBDA (Window) (* Gaska "8-Sep-88 08:04")

(* * Fill arbitrary shaped regions in the bitmap with a user specified shade)

(LET* ((Shade (EditBitmapGetShade)))
  (if Shade
    then (EditBitmapFillRegion Window Shade)))
```

(EditBitmapGetShade

```
[LAMBDA NIL (* Gaska "23-Sep-88 10:00")

(* * Get a shade)

(LET* [(Texture (MENU (create MENU
  ITEMS _
  [APPEND (for Shade in EditBitmapListOfTextures
    collect (LIST (LET ((Bitmap (BITMAPCREATE 120 16)))
      (BITBLT NIL NIL NIL Bitmap NIL NIL 120 NIL
        'TEXTURE
        'REPLACE Shade)
      Bitmap)
      Shade))
    ' ("4 X 4 SHADE" NIL "Other Shade")
    ("16 X 16 SHADE" NIL "Other Shade")
  MENUBORDERSIZE _ 1
  MENUOUTLINESIZE _ 2
  MENUFONT _ EditBitmapMenuFont
  TITLE _ "SHADE"
  CENTERFLG _ T
  WHENSELECTEDFN _ [FUNCTION (LAMBDA (Item Menu Button)
    (LIST Item Button))
  ITEMWIDTH _ 120]
(COND
  ((STREQUAL (CAAR Texture)
    "4 X 4 SHADE")
    (LET ((NewTexture (EDITSHADE)))
      (NCONC1 EditBitmapListOfTextures NewTexture)
      NewTexture))
  ((STREQUAL (CAAR Texture)
    "16 X 16 SHADE")
    (LET ((NewTexture (EDITSHADE T)))
      (NCONC1 EditBitmapListOfTextures NewTexture)
      NewTexture))
  (T (CADAR Texture]))
```

(EditBitmapGetSizeFactor

```
[LAMBDA (Type) (* Gaska "23-Sep-88 10:00")

(* * Get shrink or expansion factors)

(GETMOUSESTATE)
(LET* [(Position (CREATEPOSITION LASTMOUSEX LASTMOUSEY))
  (Width (EditBitmapPopUpMenu '(1 2 3 4)
    (CONCAT "Width " Type " Factor"))))
  (Height (AND Width (EditBitmapPopUpMenu '(1 2 3 4)
    (CONCAT "Height " Type " Factor"))
  (if Height
    then (CONS Width Height])
```

(EditBitmapInvertBitmap

```
[LAMBDA (Window) (* Gaska "22-Sep-88 09:31")

(* * Flip the bitmap.)

(LET* ((Operation (EditBitmapPopUpMenu '(Horizontal Vertical Diagonal)
  "Invert How?"))
  (SELECTQ Operation
    (Horizontal (EditBitmapDoInversion Window 'EditBitmapBitmapInvertHorizontal))
    (Vertical (EditBitmapDoInversion Window 'EditBitmapBitmapInvertVertical))
    (Diagonal (EditBitmapDoInversion Window 'EditBitmapBitmapInvertDiagonal))
    NIL))
```

(EditBitmapInvertColor

[LAMBDA (Window)

(* Gaska "26-Sep-88 08:13")

(* * Invert the color of the bitmap)

```
(BITBLT (WINDOWPROP Window 'Bitmap)
  NIL NIL (WINDOWPROP Window 'Bitmap)
  NIL NIL NIL NIL 'INVERT)
(EditBitmapDisplayBitmap Window)]
```

(EditBitmapInvertColorRegion

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:09")

(* * Invert the color of a region)

```
(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Select the region to be inverted")
  (LET* ((Region (EditBitmapGetWindowRegion Window (WINDOWPROP Window 'RegionGrid)
    EditBitmapMinRegionSize EditBitmapMinRegionSize)))
    (if Region
      then (BLTSHADE BLACKSHADE (WINDOWPROP Window 'Bitmap)
        (fetch LEFT of Region)
        (fetch BOTTOM of Region)
        (fetch WIDTH of Region)
        (fetch HEIGHT of Region)
        'INVERT)
        (EditBitmapDisplayBitmap Window))))))
```

(EditBitmapMakeExact

[LAMBDA (Window)

(* Gaska "23-Sep-88 10:02")

(* * Make the bitmap smaller by specifying a region within the window)

```
(LET* [(Position (CREATEPOSITION LASTMOUSEX LASTMOUSEY))
  (NewXOrigin (READNUM "X Origin" Position EditBitmapMenuFont 0 (IQUOTIENT (WINDOWPROP Window
    'WIDTH)
    2)
    NIL T))
  (NewYOrigin (AND NewXOrigin (READNUM "Y Origin" Position EditBitmapMenuFont 0
    (IQUOTIENT (WINDOWPROP Window 'HEIGHT)
    2)
    NIL T)))
  (NewWidth (AND NewYOrigin (READNUM "Width" Position EditBitmapMenuFont EditBitmapMinSize
    (WINDOWPROP Window 'WIDTH)
    NIL T)))
  (NewHeight (AND NewWidth (READNUM "Height" Position EditBitmapMenuFont EditBitmapMinSize
    (WINDOWPROP Window 'HEIGHT)
    NIL T))]
  (if NewHeight
    then (EditBitmapChangeBitmapSize Window (EditBitmapCopyBitmap (WINDOWPROP Window 'Bitmap)
      NewXOrigin NewYOrigin NewWidth NewHeight]))
```

(EditBitmapMakeSmaller

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:15")

(* * Make the bitmap smaller by specifying a region within the window)

```
(EditBitmapChangeBitmapSize Window (EditBitmapSmaller Window (WINDOWPROP Window 'Bitmap)
  (WINDOWPROP Window 'RegionGrid)
  EditBitmapMinSize EditBitmapMinSize))
```

(EditBitmapMask

[LAMBDA (Window)

(* Gaska "1-Sep-88 15:42")

(* * Make a "cookie cutter" mask as used in ICONW)

```
(EditBitmapChangeBitmapSizeMaybe Window (RESETLST
  (RESETSAVE (CURSOR WAITINGCURSOR))
  (EditBitmapMakeMask (WINDOWPROP Window 'Bitmap))))
```

(EditBitmapMoveRectangularRegion

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:12")

(* * Move a region)

```
(if (EditBitmapMoveRegion Window (WINDOWPROP Window 'Bitmap)
  (WINDOWPROP Window 'RegionGrid)
  (WINDOWPROP Window 'Grid)
  EditBitmapMinRegionSize EditBitmapMinRegionSize)
  then (EditBitmapDisplayBitmap Window])
```

(EditBitmapPaintWithAirbrush

[LAMBDA (Window)

(* Gaska "28-Oct-88 12:40")

(* * Allow the user to paint on a bitmap with an airbrush using the Bitmap Editor paint facility)

```
(EditBitmapAirbrushPaint Window (WINDOWPROP Window 'Bitmap)
  (WINDOWPROP Window 'BrushShape)
  (WINDOWPROP Window 'AirBrushSize)
  NIL
  (WINDOWPROP Window 'Operation)
  (WINDOWPROP Window 'AirBrushSpeed)
  EditBitmapAirbrushTimerIntervals EditBitmapMenuFont)
(EditBitmapDisplayBitmap Window)]
```

(EditBitmapPaintWithBitmap

[LAMBDA (Window)

(* Gaska " 7-Sep-88 15:10")

(* * Allow the user to paint on a bitmap with a pattern using the Bitmap Editor paint facility)

```
(LET ((Pattern (EditBitmapGetPattern Window)))
  (if Pattern
    then (EditBitmapPaintWindowWithBitmap Window (WINDOWPROP Window 'Bitmap)
      Pattern
      (WINDOWPROP Window 'Operation)
      EditBitmapMenuFont])
```

(EditBitmapPaintWithBrush

[LAMBDA (Window)

(* Gaska " 7-Sep-88 15:11")

(* * Allow the user to paint on a bitmap with a brush using the Bitmap Editor paint facility)

```
(EditBitmapPaintWindow Window (WINDOWPROP Window 'Bitmap)
  (WINDOWPROP Window 'BrushShape)
  (WINDOWPROP Window 'PaintBrushSize)
  (WINDOWPROP Window 'Shade)
  (WINDOWPROP Window 'Operation)
  EditBitmapMenuFont])
```

(EditBitmapPatternArray

[LAMBDA (EditorWindow)

(* Gaska " 6-Oct-88 09:43")

(* * Menu for specifying pattern array values)

```
(LET* [(Pattern (EditBitmapGetPattern EditorWindow))
  (Position (AND Pattern (EditBitmapGetPosition EditorWindow "Select lower left corner of array."))
  (if Position
    then (LET* [(Width (IPLUS (STRINGWIDTH "Number In X" EditBitmapMenuFont)
      5))
      (Menus (for Attribute in '(Delta% Y |Number In Y| Delta% X |Number In X| Origin% Y
        Origin% X)
        collect (EditBitmapSetPatternMenu Attribute Width)))
      (InitialValues (LIST (BITMAPHEIGHT Pattern)
        1
        (BITMAPWIDTH Pattern)
        1
        (CDR Position)
        (CAR Position)))
      (MenuHeight (fetch IMAGEHEIGHT of (CAR Menus)))
      (Window (LET NIL (GETMOUSESTATE)
        (CREATEW (CREATEREGION LASTMOUSEX LASTMOUSEY
          (WIDTHIFWINDOW [IPLUS 50
            (fetch IMAGEWIDTH
              of (for Menu in Menus
                largest (fetch IMAGEWIDTH
                  of Menu]
            4)
            (HEIGHTIFWINDOW (IPLUS (ITIMES MenuHeight 7)
              32)
            T 4))
          "Pattern Attributes" NIL T]
        (DSPFONT EditBitmapMenuFont Window)
        (for Menu in Menus as Count from 0 as Attribute
          in '(Delta% Y |Number In Y| Delta% X |Number In X| Origin% Y Origin% X) as Initial
          in InitialValues as MinValue in '(0 1 0 1 0 0)
          do (WINDOWPROP Window Attribute Initial)
            (WINDOWPROP Window (PACK* Attribute 'Min)
              MinValue)
            (ADDMENU Menu Window (CREATEPOSITION 4 (ITIMES (IPLUS MenuHeight 5)
              Count)))
            (LET ((Region (CREATEREGION (IPLUS (fetch RIGHT of (MENUREGION Menu))
              5)
              (fetch BOTTOM of (MENUREGION Menu))
              40 MenuHeight)))
```



```

(DSPFILL Region BLACKSHADE 'REPLACE Window)
(DSPFILL (CREATEREGION (IPLUS (fetch LEFT of Region)
                               2)
                        (IPLUS (fetch BOTTOM of Region)
                               2)
                        (IDIFFERENCE (fetch WIDTH of Region)
                                       4)
                        (IDIFFERENCE MenuHeight 4)))
        WHITESHADE
        'REPLACE Window)
(CENTERPRINTINREGION Initial Region Window)))
(ADDMENU [create MENU
          ITEMS _ ' (Done)
          MENUBORDERSIZE _ 1
          MENUFONT _ EditBitmapMenuFont
          ITEMWIDTH _ (fetch ITEMWIDTH of (CAR Menues))
          CENTERFLG _ T
          WHENSELECTEDFN _ (FUNCTION (LAMBDA (Item Menu Button)
                                     (EditBitmapSetPatternDone Menu]
Window
  (CREATEPOSITION 4 (ITIMES (IPLUS MenuHeight 5)
                           6))
  T)
(WINDOWPROP Window 'EditorWindow EditorWindow)
(WINDOWPROP Window 'Pattern Pattern)
(OPENW Window])

```

(EditBitmapPixelEditRegion

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:13")

(* * Edit a region)

```

(if (EditBitmapEditRegion Window (WINDOWPROP Window 'Bitmap)
    'EDITBM
    (WINDOWPROP Window 'RegionGrid)
    (WINDOWPROP Window 'Grid)
    EditBitmapMinRegionSize EditBitmapMinRegionSize)
    then (EditBitmapDisplayBitmap Window])

```

(EditBitmapPixelEditWindow

[LAMBDA (Window)

(* Gaska " 7-Sep-88 11:16")

(* * Invoke the pixel editor)

```

(if [BITMAPP (EDITBM (WINDOWPROP Window 'Bitmap)
    then (EditBitmapDisplayBitmap Window])

```

(EditBitmapPlacePattern

[LAMBDA (Window)

(* Gaska "15-Nov-88 11:11")

(* * Place a pattern)

```

(LET* ((Pattern (EditBitmapGetPattern Window)))
  (if Pattern
    then (RESETLST
          (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
          (EditBitmapMessage Window '("Place the pattern within the window." "Click outside the
                                     window to terminate."))
          (bind Dirty? Position while [SETQ Position (EditBitmapPlaceBitmap Window Pattern NIL
                                                         (WINDOWPROP Window 'Grid]
          do (if Position
              then (BITBLT Pattern NIL NIL (WINDOWPROP Window 'Bitmap)
                          (CAR Position)
                          (CDR Position)
                          NIL NIL 'INPUT (WINDOWPROP Window 'Operation))
              (BITBLT Pattern NIL NIL Window (CAR Position)
                          (CDR Position)
                          NIL NIL 'INPUT (WINDOWPROP Window 'Operation))
              (SETQ Dirty? T)
              (UNTILMOUSESTATE UP))
          finally (if Dirty?
                  then (EditBitmapDisplayBitmap Window))
                  (RETURN Dirty?))))))

```

(EditBitmapResetBitmap

[LAMBDA (Window)

(* Gaska " 3-Oct-88 08:30")

(* * Restore bitmap to the original state)

```

(if (EditBitmapConfirmDiscard Window)
    then (EditBitmapChangeBitmapSizeMaybe Window (EditBitmapCopyBitmap (WINDOWPROP Window 'OriginalBitmap])

```

(EditBitmapRotateBitmap

[LAMBDA (Window)

(* Gaska "21-Sep-88 15:57")

(* * Rotate the bitmap 90 degrees.)

```
(LET ((Operation (EditBitmapPopUpMenu '(|Left 90 Degrees| |Right 90 Degrees| Arbitrary% Angle)
                                     "Rotate How?"))))
(SELECTQ Operation
  (|Left 90 Degrees|
    (EditBitmapDoRotation Window 'EditBitmapBitmapRotate 'Left))
  (|Right 90 Degrees|
    (EditBitmapDoRotation Window 'EditBitmapBitmapRotate 'Right))
  (Arbitrary% Angle
    (LET ((Angle (READNUM "Angle" NIL EditBitmapMessageFont -360 360)))
      (if Angle
        then (EditBitmapDoRotation Window 'EditBitmapBitmapRotateArbitrary Angle))))
  NIL])
```

(EditBitmapSaveBitmap

[LAMBDA (Window)

(* Gaska "26-Sep-88 08:14")

(* * Save current bitmap for UNDO)

```
(LET* ((Bitmap (WINDOWPROP Window 'Bitmap))
      (Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap)))
  (BITBLT Bitmap NIL NIL (WINDOWPROP Window 'SavedBitmap))
  (WINDOWPROP Window 'SavedBitmapWidth Width)
  (WINDOWPROP Window 'SavedBitmapHeight Height])
```

(EditBitmapSetPatternAttribute

[LAMBDA (Menu Attribute)

(* Gaska "6-Oct-88 08:49")

(* * Set the pattern attribute)

```
(SHADEITEM Attribute Menu GRAYSHADE)
(LET ((Window (WFROMMENU Menu)))
  (WINDOWPROP Window Attribute (READNUM Attribute NIL EditBitmapMenuFont (WINDOWPROP Window
                                                                                   (PACK* Attribute
                                                                                   'Min)))

    (LET [(Region (CREATEREGION (IPLUS (fetch RIGHT of (MENUMREGION Menu))
                                         1000 NIL T T))
      (IPLUS (fetch BOTTOM of (MENUMREGION Menu))
              7)
      (IPLUS (fetch IMAGEHEIGHT of Menu)
              2)
      (IDIFFERENCE (fetch IMAGEHEIGHT of Menu)
                    36)
      (DSPFILL Region WHITESHADE 'REPLACE Window)
      (CENTERPRINTINREGION (WINDOWPROP Window Attribute)
                            Region Window))
      (SHADEITEM Attribute Menu WHITESHADE)])
```

(EditBitmapSetPatternDone

[LAMBDA (Menu)

(* Gaska "6-Oct-88 08:44")

(* * Pattern array parameters are done)

```
(LET* [(Window (WFROMMENU Menu))
      (EditorWindow (WINDOWPROP Window 'EditorWindow))
      (Bitmap (WINDOWPROP EditorWindow 'Bitmap))
      (OriginX (WINDOWPROP Window 'Origin% X))
      (OriginY (WINDOWPROP Window 'Origin% Y))
      (NumberInX (WINDOWPROP Window 'Number In X))
      (DeltaX (WINDOWPROP Window 'Delta% X))
      (NumberInY (WINDOWPROP Window 'Number In Y))
      (DeltaY (WINDOWPROP Window 'Delta% Y))
      (Pattern (WINDOWPROP Window 'Pattern))
      (WINDOWPROP Window 'MENU NIL)
      (WINDOWPROP Window 'EditorWindow NIL)
      (WINDOWPROP Window 'Pattern NIL)
      (CLOSEW Window)
      (if (AND (IGREATERP NumberInX 0)
                (IGREATERP DeltaX 0)
                (IGREATERP NumberInY 0)
                (IGREATERP DeltaY 0))
        then [for Y from 0 to (ITIMES (SUB1 NumberInY)
                                         DeltaY)
              by DeltaY do (for x from 0 to (ITIMES (SUB1 NumberInX)
                                                         DeltaX)
                            by DeltaX do (BITBLT Pattern NIL NIL Bitmap (IPLUS OriginX X)
                                             (IPLUS OriginY Y)
                                             NIL NIL 'INPUT (WINDOWPROP EditorWindow 'Operation))
                                             (BITBLT Pattern NIL NIL EditorWindow (IPLUS OriginX X))
```

```

(IPLUS OriginY Y)
NIL NIL 'INPUT (WINDOWPROP EditorWindow 'Operation]
(EditBitmapDisplayBitmap EditorWindow])

```

(EditBitmapSetPatternMenu

[LAMBDA (Attribute Width)

(* Gaska "5-Oct-88 15:03")

(* * Create one of the pattern attribute menus)

```

(create MENU
  ITEMS _ (LIST Attribute)
  MENUBORDERSIZE _ 1
  MENUFONT _ EditBitmapMenuFont
  ITEMWIDTH _ Width
  CENTERFLG _ T
  WHENSELECTEDFN _ (FUNCTION (LAMBDA (Item Menu Button)
    (EditBitmapSetPatternAttribute Menu Item))

```

(EditBitmapSetSide

[LAMBDA (Menu Side)

(* Gaska "2-Sep-88 10:34")

(* * Set the number of bits to remove from a side)

```

(SHADEITEM Side Menu GRAYSHADE)
(LET ((Window (WFROMMENU Menu)))
  (WINDOWPROP Window Side (READNUM (LIST "Number of Bits to" (CONCAT "Remove from " Side))
    NIL EditBitmapMenuFont 0
    (SUB1 (IQUOTIENT (if (MEMBER Side '(Left Right))
      then (WINDOWPROP Window 'WIDTH)
      else (WINDOWPROP Window 'HEIGHT))
    2))
    NIL T T))
  (LET [(Region (CREATEREGION (IPLUS (fetch RIGHT of (MENUREGION Menu))
    7)
    (IPLUS (fetch BOTTOM of (MENUREGION Menu))
    2)
    36
    (IDIFFERENCE (fetch IMAGEHEIGHT of Menu)
    4]
    (DSPFILL Region WHITESHADE 'REPLACE Window)
    (CENTERPRINTINREGION (WINDOWPROP Window Side)
      Region Window))
    (SHADEITEM Side Menu WHITESHADE)])

```

(EditBitmapSetSideDone

[LAMBDA (Menu)

(* Gaska "30-Aug-88 14:12")

(* * Side parameters are done)

```

(LET* [(Window (WFROMMENU Menu))
  (EditorWindow (WINDOWPROP Window 'EditorWindow))
  (Bitmap (WINDOWPROP EditorWindow 'Bitmap))
  (Left (WINDOWPROP Window 'Left))
  (Bottom (WINDOWPROP Window 'Bottom))
  (Right (WINDOWPROP Window 'Right))
  (Top (WINDOWPROP Window 'Top))
  (WINDOWPROP Window 'MENU NIL)
  (WINDOWPROP Window 'EditorWindow NIL)
  (CLOSEW Window)
  (EditBitmapChangeBitmapSize EditorWindow (EditBitmapCopyBitmap Bitmap Left Bottom
    (IDIFFERENCE (IDIFFERENCE (BITMAPWIDTH Bitmap)
      Left)
      Right)
    (IDIFFERENCE (IDIFFERENCE (BITMAPHEIGHT Bitmap)
      Top)
      Bottom)])

```

(EditBitmapSetSideMenu

[LAMBDA (Side Width)

(* Gaska "30-Aug-88 11:25")

(* * Create one of the side menus)

```

(create MENU
  ITEMS _ (LIST Side)
  MENUBORDERSIZE _ 1
  MENUFONT _ EditBitmapMenuFont
  ITEMWIDTH _ Width
  CENTERFLG _ T
  WHENSELECTEDFN _ (FUNCTION (LAMBDA (Item Menu Button)
    (EditBitmapSetSide Menu Item))

```

(EditBitmapShiftBitmap

```
[LAMBDA (Window) (* Gaska "23-Sep-88 10:06")

(* * Shift the bitmap left, right, up, or down.)

(LET* [(Operation (EditBitmapPopUpMenu ' (Left Right Down Up)
      "Shift How?"))
      (Bits (AND Operation (READNUM (LIST "Number of bits" (CONCAT "to Shift " Operation))
      NIL EditBitmapMenuFont 0 200 NIL T NIL 3])
      (if Bits
        then (EditBitmapChangeBitmapSize Window (SELECTQ Operation
          (Left (EditBitmapBitmapShift (WINDOWPROP Window
            'Bitmap)
              (IMINUS Bits)
              0))
          (Right (EditBitmapBitmapShift (WINDOWPROP Window
            'Bitmap)
              Bits 0))
          (Down (EditBitmapBitmapShift (WINDOWPROP Window
            'Bitmap)
              0
              (IMINUS Bits)))
          (Up (EditBitmapBitmapShift (WINDOWPROP Window
            'Bitmap)
              0 Bits))
          NIL))
        NIL])]
```

(EditBitmapShrinkBitmap

```
[LAMBDA (Window) (* Gaska "7-Sep-88 11:18")

(* * Shrink the bitmap)

(LET* ((Factors (EditBitmapGetSizeFactor "Shrink")))
  (if Factors
    then (RESETLST
      (RESETSAVE (CURSOR WAITINGCURSOR))
      (EditBitmapChangeBitmapSize Window (SHRINKBITMAP (WINDOWPROP Window 'Bitmap)
        (CAR Factors)
        (CDR Factors))))))]
```

(EditBitmapTessellate

```
[LAMBDA (Window Region Pattern Clipped?) (* Gaska "26-Sep-88 08:38")

(* * Tessellate the bitmap with a pattern)

(LET* ((PatternWidth (BITMAPWIDTH Pattern))
      (PatternHeight (BITMAPHEIGHT Pattern))
      (Width (IMIN (ITIMES PatternWidth (IPLUS (IQUOTIENT (fetch WIDTH of Region)
        PatternWidth)
        (OR (AND Clipped? 0)
        1)))
        (fetch WIDTH of Region)))
      (Height (IMIN (ITIMES PatternHeight (IPLUS (IQUOTIENT (fetch HEIGHT of Region)
        PatternHeight)
        (OR (AND Clipped? 0)
        1)))
        (fetch HEIGHT of Region)))
      (Bitmap (WINDOWPROP Window 'Bitmap))
      (TempBitmap (BITMAPCREATE Width Height)))
  (if (AND (IGREATERP Width 0)
    (IGREATERP Height 0))
    then (RESETLST
      (RESETSAVE (CURSOR WAITINGCURSOR))
      (for Y from 0 to Height by PatternHeight
        do (for X from 0 to Width by PatternWidth
          do (BITBLT Pattern 0 0 TempBitmap X Y PatternWidth PatternHeight)))
      (BITBLT TempBitmap 0 0 Bitmap (fetch LEFT of Region)
        (fetch BOTTOM of Region)
        (fetch WIDTH of Region)
        (fetch HEIGHT of Region)
        'INPUT
        (WINDOWPROP Window 'Operation))
      (EditBitmapDisplayBitmap Window))
    else (EditBitmapMessage Window "Region too small to hold entire pattern."))]
```

(EditBitmapTessellateBitmap

```
[LAMBDA (Window) (* Gaska "8-Sep-88 12:48")

(* * Tessellate the bitmap with a pattern)

(LET* ((Pattern (EditBitmapGetPattern Window))
      (if Pattern
        then (EditBitmapTessellate Window (CREATEREGION 0 0 (WINDOWPROP Window 'WIDTH)
          (WINDOWPROP Window 'HEIGHT))
          Pattern)
```

```

(EditBitmapPopupMenu ' ((Yes T)
                      (No NIL))
"Clip to Integer Number?"))

```

(EditBitmapTessellateRegion

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:09")

(* * Tessellate a region within the bitmap with a pattern)

```

(LET* [(Region (RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Select the region to be tessellated")
  (EditBitmapGetWindowRegion Window (WINDOWPROP Window 'RegionGrid)
    EditBitmapMinRegionSize EditBitmapMinRegionSize)))
  (Pattern (AND Region (EditBitmapGetPattern Window)
  (if Pattern
    then (EditBitmapTessellate Window Region Pattern (EditBitmapPopupMenu ' ((Yes T)
                                                                    (No NIL))
                                                                    "Clip to Integer Number?"))

```

(EditBitmapTrimBitmap

[LAMBDA (Window)

(* Gaska "25-Aug-88 08:25")

(* * Remove the white space from the edges of the bitmap)

```

(EditBitmapChangeBitmapSize Window (RESETLST
  (RESETSAVE (CURSOR WAITINGCURSOR))
  (EditBitmapTrim (WINDOWPROP Window 'Bitmap)))]

```

(EditBitmapTrimSides

[LAMBDA (EditorWindow)

(* Gaska "30-Aug-88 14:10")

(* * Menu for setting bits to remove from each side)

```

(LET* [(Width (IPLUS (STRINGWIDTH "Bottom" EditBitmapMenuFont)
  5))
  (Menues (for Side in ' (Top Right Left Bottom) collect (EditBitmapSetSideMenu Side Width)))
  (MenuHeight (fetch IMAGEHEIGHT of (CAR Menues)))
  (Window (LET NIL (GETMOUSESTATE)
    (CREATEW (CREATEREGION LASTMOUSEX LASTMOUSEY
      (WIDTHIFWINDOW [IPLUS 50 (fetch IMAGEWIDTH
        of (for Menu in Menues
          largest (fetch IMAGEWIDTH of Menu]
        4)
      (HEIGHTIFWINDOW (IPLUS (ITIMES MenuHeight 5)
        20)
      T 4))
    "Bits to Remove" NIL T)
  (DSPFONT EditBitmapMenuFont Window)
  (for Menu in Menues as Count from 0 as Side in ' (Top Right Left Bottom)
    do (WINDOWPROP Window Side 0)
      (ADDMENU Menu Window (CREATEPOSITION 4 (ITIMES (IPLUS MenuHeight 5)
        Count)))
      (LET ((Region (CREATEREGION (IPLUS (fetch RIGHT of (MENUREGION Menu))
        5)
        (fetch BOTTOM of (MENUREGION Menu))
        40 MenuHeight)))
        (DSPFILL Region BLACKSHADE 'REPLACE Window)
        (DSPFILL (CREATEREGION (IPLUS (fetch LEFT of Region)
        2)
        (IPLUS (fetch BOTTOM of Region)
        2)
        (IDIFFERENCE (fetch WIDTH of Region)
        4)
        (IDIFFERENCE MenuHeight 4))
        WHITESHADE
        'REPLACE Window)
        (CENTERPRINTINREGION 0 Region Window)))
      (ADDMENU [create MENU
        ITEMS _ ' (Done)
        MENUBORDERSIZE _ 1
        MENUFONT _ EditBitmapMenuFont
        ITEMWIDTH _ (fetch ITEMWIDTH of (CAR Menues))
        CENTERFLG _ T
        WHENSELECTEDFN _ (FUNCTION (LAMBDA (Item Menu Button)
          (EditBitmapSetSideDone Menu]
        Window
        (CREATEPOSITION 4 (ITIMES (IPLUS MenuHeight 5)
        4))
        T)
      (WINDOWPROP Window 'EditorWindow EditorWindow)
      (OPENW Window)]

```

(EditBitmapUndo

[LAMBDA (Window)

(* Gaska "26-Sep-88 08:14")

(* * UNDO last edit operation)

```

(WINDOWPROP Window 'Dirty (WINDOWPROP Window 'LastDirtyState))
(if [AND [EQ (WINDOWPROP Window 'SavedBitmapWidth)
            (BITMAPWIDTH (WINDOWPROP Window 'Bitmap)
            (EQ (WINDOWPROP Window 'SavedBitmapHeight)
            (BITMAPHEIGHT (WINDOWPROP Window 'Bitmap)
            then (BITBLT (WINDOWPROP Window 'SavedBitmap)
                        0 0 (WINDOWPROP Window 'Bitmap)
                        0 0 (WINDOWPROP Window 'SavedBitmapWidth)
                        (WINDOWPROP Window 'SavedBitmapHeight))
            (REDISPLAYW Window)
            else (EditBitmapChangeBitmapSize Window (EditBitmapCopyBitmap (WINDOWPROP Window 'SavedBitmap)
                        0 0 (WINDOWPROP Window 'SavedBitmapWidth)
                        (WINDOWPROP Window 'SavedBitmapHeight]))

```

)

(* * Averaging functions)

(DEFINEQ

(EditBitmapAverageBitInBitmap

[LAMBDA (Bitmap X Y Weights Threshold)

(* Gaska "20-Sep-88 13:19")

(* * Average a bit in a bitmap according to a weighted array and threshold)

```

(if (IGEQ (for Weight in Weights as XOffset in '(-1 0 1 -1 0 1 -1 0 1) as YOffset
          in '(1 1 1 0 0 0 -1 -1 -1) sum (if (ZEROP (BITMAPBIT Bitmap (IPLUS X XOffset)
                                                    (IPLUS Y YOffset)))
          then 0
          else Weight))
    Threshold)
    then 1
    else 0))

```

(EditBitmapAveraging

[LAMBDA (Window)

(* Gaska "26-Oct-88 10:08")

(* * Perform averging on the bitmap. Use the default averaging information or let the user supply a name of a list)

```

(LET* [(AveragingInfo (EditBitmapGetAveragingInfo Window))
      (Block? (AND AveragingInfo (EditBitmapBlock?))
      (if AveragingInfo
          then (RESETLST
                (RESETSAVE (CURSOR (if Block?
                                      then T
                                      else WAITINGCURSOR))))
          (if Block?
              then (SPAWN.MOUSE))
          (EditBitmapMessage Window "This may take a LONG time!")
          (LET ((NewBitmap (EditBitmapBitmapAverage (WINDOWPROP Window 'Bitmap)
                                                    (CAR AveragingInfo)
                                                    (CADR AveragingInfo)
                                                    Block? Window)))
              (EditBitmapMessage Window "Averaging complete.")
              (BITBLT NewBitmap NIL NIL (WINDOWPROP Window 'Bitmap))
              (EditBitmapDisplayBitmap Window
              T)))]

```

(EditBitmapBitmapAverage

[LAMBDA (Bitmap Weights Threshold Block FeedbackWindow)

(* Gaska "21-Sep-88 16:16")

(* * Create a new bitmap using an averaging array and threshold upon the original bitmap)

```

(LET* ((Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (NewBitmap (BITMAPCREATE Width Height))
      [bind NewBit for Y from 0 to (SUB1 Height)
          do (for X from 0 to (SUB1 Width) do (SETQ NewBit (EditBitmapAverageBitInBitmap Bitmap X Y Weights
                                                    Threshold))
              (BITMAPBIT NewBitmap X Y NewBit)
              (if FeedbackWindow
                  then (BITMAPBIT FeedbackWindow X Y NewBit))
              (if Block
                  then (BLOCK Block]
          NewBitmap])

```

(EditBitmapGetAveragingInfo

[LAMBDA (Window)

(* Gaska "20-Sep-88 14:33")

```
(* * Auto-Save functions)

(DEFINEQ

(EditBitmapAutoSave
 [LAMBDA (Window)

    (* * Automatically save the bitmap if conditions are correct)

    (if [AND (WINDOWPROP Window 'AutoSave)
             (IGEQ (IDIFFERENCE (IDATE)
                                (WINDOWPROP Window 'AutoSaveLastSave))
              (WINDOWPROP Window 'AutoSaveDeltaTime)
             ]
      then (OBTAIN.MONITORLOCK (WINDOWPROP Window 'Monitor))
            [LET* [(File (WINDOWPROP Window 'AutoSaveFileName)
                        (EditBitmapMessage Window "Saving bitmap")
                        (RESETLST
                         (RESETSAVE (CURSOR WAITINGCURSOR))
                         (RESETSAVE NIL (LIST 'CLOSEF (MKATOM File))))
                  [LET [(Stream (OPENSTREAM File 'OUTPUT)
                                (EditBitmapWriteBitmap Stream (WINDOWPROP Window 'Bitmap)
                                                         (WINDOWPROP Window 'AutoSaveLastSave (IDATE))
                                                         (EditBitmapMessage Window (CONCAT "Bitmap has been saved on file: " (FULLNAME File)))]
                      (RELEASE.MONITORLOCK (WINDOWPROP Window 'Monitor))]]
            (GASKA "18-Sep-88 12:44")
            )
    )
)
```

[illegible]

```

      (if (WINDOWPROP Window 'AutoSave)
          then (WINDOWPROP Window 'AutoSaveLastSave (IDATE)))
    else (EditBitmapMessage Window "An Auto-Save file name must be specified.")])

```

(EditBitmapSetAutoSaveFile

```

[LAMBDA (Window) (* Gaska "18-Sep-88 11:33")

```

```

  (* * Specify an auto-save file name)

```

```

  (LET ((FileName (EditBitmapGetFileName Window)))
    (WINDOWPROP Window 'AutoSaveFileName FileName)
    FileName])

```

(EditBitmapSetAutoSaveInterval

```

[LAMBDA (Window) (* Gaska "16-Sep-88 16:44")

```

```

  (* * Set the auto-save interval)

```

```

  (WINDOWPROP Window 'AutoSaveDeltaTime (ITIMES (READNUM (LIST "Auto-Save interval in minutes"
                                                                (CONCAT "Current value is: "
                                                                (IQUOTIENT (WINDOWPROP Window
                                                                'AutoSaveDeltaTime)
                                                                60))))
    NIL EditBitmapMenuFont 1 NIL NIL T T 4)
  60])

```

(EditBitmapShowAutoSave

```

[LAMBDA (Window) (* Gaska "16-Sep-88 16:39")

```

```

  (* * Display the auto-save parameters)

```

```

  (EditBitmapMessage Window (LIST (CONCAT "Auto-Save - " (if (WINDOWPROP Window 'AutoSave)
                                                                then "ON"
                                                                else "OFF"))
    (CONCAT "Auto-Save File Name - " (OR (WINDOWPROP Window 'AutoSaveFileName)
    " ")))
    (CONCAT "Auto-Save Interval - " (IQUOTIENT (WINDOWPROP Window
    'AutoSaveDeltaTime)
    60)
    " minutes"]])

```

```

)

```

```

  (* * Magnification functions)

```

```

(DEFINEQ

```

(EditBitmapMagnify

```

[LAMBDA (Window Bitmap ButtonEventFn PromptMessage) (* Gaska "2-Nov-88 10:59")

```

```

  (* * Magnify the bitmap)

```

```

  (LET [(MagnifyWindow (LET NIL (RESETLST
    (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
    (EditBitmapMessage Window "Place the magnifier display window.")
    (CREATEW (GETBOXREGION (WIDTHIFWINDOW 128 4)
    (HEIGHTIFWINDOW 148 T 4))
    "8X Magnification"))])
    (DRAWLINE 0 129 128 129 2 'PAINT MagnifyWindow)
    (DSPFONT ' (GACHA 12 BRR)
    MagnifyWindow)
    (if PromptMessage
      then (EditBitmapMessage Window PromptMessage))
    (WINDOWPROP MagnifyWindow 'MainWindow Window)
    (WINDOWPROP MagnifyWindow 'MagnifySourceBitmap Bitmap)
    (WINDOWPROP MagnifyWindow 'TinyBitmap (BITMAPCREATE 16 16))
    (WINDOWPROP MagnifyWindow 'MagnifiedBitmap (BITMAPCREATE 128 128))
    (WINDOWADDPROP MagnifyWindow 'CLOSEFN 'EditBitmapMagnifyClose T)
    (WINDOWADDPROP MagnifyWindow 'CLOSEFN 'DON'T T)
    (WINDOWPROP Window 'MagnifyWindow MagnifyWindow)
    (WINDOWPROP MagnifyWindow 'RESHAPEFN 'DON'T)
    (WINDOWPROP MagnifyWindow 'SHRINKFN 'DON'T)
    [WINDOWPROP Window 'CURSORINFN (FUNCTION (LAMBDA (Window)
    (CURSOR EditBitmapMagnifyCursor)
    (DRAWLINE 0 129 128 129 2 'PAINT (WINDOWPROP Window
    'MagnifyWindow)

    [WINDOWPROP Window 'CURSOROUTFN (FUNCTION (LAMBDA (Window)
    (CLEARW (WINDOWPROP Window 'MagnifyWindow))
    (CURSOR T)

    (WINDOWPROP Window 'CURSORMOVEDFN 'EditBitmapMagnifyMoved)
    (WINDOWPROP Window 'BUTTONEVENTFN ButtonEventFn)
    (WINDOWPROP Window 'RIGHTBUTTONFN ButtonEventFn)
    MagnifyWindow])

```


(EditBitmapMagnifyBitmap

[LAMBDA (Source Destination)

(* Gaska "29-Sep-88 13:07")

(* * Magnify a bitmap. Stolen from MAGNIFYW package.)

```
(bind Word (SourceBase _ (FETCHFIELD ' (BITMAP 0 POINTER)
                                   Source))
      (DestinationBase _ (FETCHFIELD ' (BITMAP 0 POINTER)
                                   Destination))
for Byte from 0 to 15 as SourceByte from 0 by (CONSTANT (ROT 1 7 16))
do (SETQ Word (\GETBASE SourceBase Byte))
  (for Bit from SourceByte to (IPLUS SourceByte 15) do (\PUTBASEBYTE DestinationBase Bit
                                                         (COND
                  ((BITTEST Word (MASK.1'S 15 1))
                   (MASK.1'S 0 8))
                  (T 0)))
                  (SETQ Word (ROT Word 1 16))))
  (for DestinationByte from (IPLUS SourceByte 16) to (IPLUS SourceByte (CONSTANT (ROT 7 4 16)))
    by 16 do (\MOVEBYTES DestinationBase SourceByte DestinationBase DestinationByte 16]))
```

(EditBitmapMagnifyChange

[LAMBDA (Window X Y NewState)

(* Gaska "30-Sep-88 08:49")

(* * Change the state of the magnified bitmap under the cursor)

```
(LET [(MagnifyWindow (WINDOWPROP Window 'MagnifyWindow)
 (BITMAPBIT (WINDOWPROP MagnifyWindow 'MagnifySourceBitmap)
             X Y NewState)
 (BITMAPBIT Window X Y NewState)
 (BITMAPBIT (WINDOWPROP MagnifyWindow 'TinyBitmap)
             7 7 NewState)
 (EditBitmapMagnifyBitmap (WINDOWPROP MagnifyWindow 'TinyBitmap)
 (WINDOWPROP MagnifyWindow 'MagnifiedBitmap))
 (BITBLT (WINDOWPROP MagnifyWindow 'MagnifiedBitmap)
          NIL NIL MagnifyWindow)
 (DRAWLINE 0 60 128 60 2 'PAINT (WINDOWPROP MagnifyWindow 'DSP))
 (DRAWLINE 60 0 60 128 2 'PAINT (WINDOWPROP MagnifyWindow 'DSP))
```

(EditBitmapMagnifyClose

[LAMBDA (Window)

(* Gaska "30-Sep-88 08:46")

(* * CloseFn for magnify window)

```
(LET [(MainWindow (WINDOWPROP Window 'MainWindow)
 (WINDOWPROP MainWindow 'CURSORINFN NIL)
 (WINDOWPROP MainWindow 'CURSOROUTFN NIL)
 (WINDOWPROP MainWindow 'CURSORMOVEDFN NIL)
 (WINDOWPROP MainWindow 'BUTTONEVENTFN NIL)
 (WINDOWPROP MainWindow 'RIGHTBUTTONFN NIL)
 (WINDOWPROP MainWindow 'MagnifyWindow NIL)
 (WINDOWPROP Window 'MainWindow NIL)
 (WINDOWPROP Window 'MagnifySourceBitmap NIL)
 (WINDOWPROP Window 'TinyBitmap NIL)
 (WINDOWPROP Window 'MagnifiedBitmap NIL)
 (CURSOR T))
```

(EditBitmapMagnifyEdit

[LAMBDA (Window)

(* Gaska "30-Sep-88 09:11")

(* * Magnify and edit the bitmap)

```
(RESETLST
 (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
 [LET [(NewBitmap (EditBitmapCopyBitmap (WINDOWPROP Window 'Bitmap)
 (ATTACHMENU [create MENU
                  ITEMS _ ' (OK ABORT)
                  MENCOLUMNS _ 2
                  MENUFONT _ EditBitmapMenuFont
                  CENTERFLG _ T
                  MENUBORDERSIZE _ 1
                  WHENSELECTEDFN _ (FUNCTION (LAMBDA (Item Menu Button)
                                              (WINDOWPROP (WINDOWPROP (WFROMMENU Menu)
                                                                    'MAINWINDOW)
                                                                    'Finished Item]
 (EditBitmapMagnify Window NewBitmap 'EditBitmapMagnifyEditButtonEventFn ' ("Left button turns
                                                                 bit on." "Right
                                                                 button turns
                                                                 bit off."))
 ' TOP
 ' JUSTIFY)
 (SPAWN.MOUSE)
 (bind Done until (SETQ Done (WINDOWPROP (WINDOWPROP Window 'MagnifyWindow)
```

```

                                'Finished))
    do (DISMISS 250) finally [if (EQ Done 'OK)
                                then (BITBLT NewBitmap NIL NIL (WINDOWPROP Window 'Bitmap)
                                (EditBitmapDisplayBitmap Window)
                                (WINDOWDELPROP (WINDOWPROP Window 'MagnifyWindow)
                                'CLOSEFN
                                'DON'T)
                                (CLOSEW (WINDOWPROP Window 'MagnifyWindow))
                                (RETURN (EQ Done 'OK)))]

```

(EditBitmapMagnifyEditButtonEventFn

[LAMBDA (Window)

(* Gaska "3-Oct-88 10:35")

(* * ButtoneventFn for magnify)

```

(GETMOUSESTATE)
(LET [[X (LASTMOUSEX (WINDOWPROP Window 'DSP)
(Y (LASTMOUSEY (WINDOWPROP Window 'DSP)
(if (INSIDEP Window X Y)
    then (COND
        ((LASTMOUSESTATE LEFT)

(* * Turn the corresponding bit on)

        (EditBitmapMagnifyChange Window X Y 1))
        ((LASTMOUSESTATE RIGHT)

(* * Turn the corresponding bit off)

        (EditBitmapMagnifyChange Window X Y 0))
        (T NIL)]

```

(EditBitmapMagnifyMoved

[LAMBDA (Window)

(* Gaska "14-Oct-88 13:22")

(* * MovedFn for magnify)

```

(LET [(MagnifyWindow (WINDOWPROP Window 'MagnifyWindow)
(BITBLT NIL NIL NIL (WINDOWPROP MagnifyWindow 'TinyBitmap)
    NIL NIL NIL NIL 'TEXTURE 'REPLACE WHITESHADE)
(BITBLT (WINDOWPROP MagnifyWindow 'MagnifySourceBitmap)
    (IDIFFERENCE (LASTMOUSEX (WINDOWPROP Window 'DSP))
    7)
    (IDIFFERENCE (LASTMOUSEY (WINDOWPROP Window 'DSP))
    7)
    (WINDOWPROP MagnifyWindow 'TinyBitmap)
    0 0 16 16)
(EditBitmapMagnifyBitmap (WINDOWPROP MagnifyWindow 'TinyBitmap)
    (WINDOWPROP MagnifyWindow 'MagnifiedBitmap))
(BITBLT (WINDOWPROP MagnifyWindow 'MagnifiedBitmap)
    NIL NIL MagnifyWindow)
(DRAWLINE 0 60 128 60 2 'PAINT (WINDOWPROP MagnifyWindow 'DSP))
(DRAWLINE 60 0 60 128 2 'PAINT (WINDOWPROP MagnifyWindow 'DSP))
(MOVETO 8 136 (WINDOWPROP MagnifyWindow 'DSP))
(PRINTOUT MagnifyWindow "X=" .I4 (LASTMOUSEX (WINDOWPROP Window 'DSP))
    ", Y=" .I4 (LASTMOUSEY (WINDOWPROP Window 'DSP))

```

(EditBitmapMagnifySelectButtonEventFn

[LAMBDA (Window)

(* Gaska "29-Sep-88 14:00")

(* * ButtoneventFn for magnify)

```

(GETMOUSESTATE)
(LET [[X (LASTMOUSEX (WINDOWPROP Window 'DSP)
(Y (LASTMOUSEY (WINDOWPROP Window 'DSP)
(COND
    ((LASTMOUSESTATE (OR LEFT RIGHT MIDDLE))

(* * Turn the corresponding bit on)

    (WINDOWPROP Window 'SelectedPixel (CONS X Y)))
    (T NIL)]

```

)

(* * Pixel follow funtions)

(DEFINEQ

(EditBitmapDrawOverPixels

[LAMBDA (Window Elements PixelList)

(* Gaska "11-Nov-88 12:34")

(* * Remove on pixels from the bitmap)

```

(LET* ([Brush (OR (AND (WINDOWPROP Window 'Patterns)
                        (EQ (EditBitmapPopupMenu ' (Pattern Brush)
                          "Draw With?")
                          'Pattern)
                        (EditBitmapGetPattern Window))
        (\GETBRUSH (LIST (WINDOWPROP Window 'BrushShape)
                          (WINDOWPROP Window 'BrushSize)
                          (Offsets (EditBitmapSelectBrushOffset Brush)))
        (RESETLST
        (RESETSAVE (CURSOR WAITINGCURSOR))
        (EditBitmapMessage Window "This might take some time.")
        (EditBitmapDrawOverConnectedPixels (WINDOWPROP Window 'Bitmap)
        Elements PixelList Window Brush (CAR Offsets)
        (CDR Offsets)
        (WINDOWPROP Window 'Operation))
        (EditBitmapMessage Window "Drawing complete.")
        (EditBitmapDisplayBitmap Window))))

```

(EditBitmapDrawOverPixelsRegion

```
[LAMBDA (Window) (* Gaska "11-Nov-88 10:03")
```

```
(* * Draw over connected pixels in the bitmap)
```

```
(EditBitmapFollowPixelsRegion Window 'EditBitmapDrawOverPixels])
```

(EditBitmapDrawOverPixelsSingle

```
[LAMBDA (Window) (* Gaska "11-Nov-88 10:04")
```

```
(* * Draw over connected pixels in the bitmap)
```

```
(EditBitmapFollowPixelsSingle Window 'EditBitmapDrawOverPixels])
```

(EditBitmapErasePixelsRegion

```
[LAMBDA (Window) (* Gaska "11-Nov-88 09:49")
```

```
(* * Remove connected pixels from the bitmap)
```

```
(EditBitmapFollowPixelsRegion Window 'EditBitmapRemovePixels])
```

(EditBitmapErasePixelsSingle

```
[LAMBDA (Window) (* Gaska "11-Nov-88 10:00")
```

```
(* * Remove connected pixels from the bitmap)
```

```
(EditBitmapFollowPixelsSingle Window 'EditBitmapRemovePixels])
```

(EditBitmapFollowPixelsMatrix

```
[LAMBDA (Window) (* Gaska "11-Nov-88 09:46")
```

```
(* * Create the follow matrix menu)
```

```

(LET* ([Positions '((-1 . 1)
                    (0 . 1)
                    (1 . 1)
                    (-1 . 0)
                    (0 . 0)
                    (1 . 0)
                    (-1 . -1)
                    (0 . -1)
                    (1 . -1)
                    (MenuItemWidth (IPLUS (STRINGWIDTH "-1,-1" EditBitmapMenuFont)
                                           20))
                    (MenuItemHeight (IPLUS (FONTHEIGHT EditBitmapMenuFont)
                                           20))
                    (MatrixMenu (create MENU
                                         ITEMS _ (for Position in Positions collect (LIST (CONCAT (CAR Position)
                                                                                               ", "
                                                                                               (CDR Position))
                                                                                               Position))
                                         ITEMWIDTH _ MenuItemWidth
                                         ITEMHEIGHT _ MenuItemHeight
                                         MENCOLUMNS _ 3
                                         MENUFONT _ EditBitmapMenuFont
                                         CENTERFLG _ T
                                         MENUBORDERSIZE _ 1
                                         WHENSELECTEDFN _ 'EditBitmapFollowPixelsMatrixSelect))
                    (DoneMenu (create MENU
                                     ITEMS _ ' (OK ABORT)
                                     MENCOLUMNS _ 2
                                     MENUFONT _ EditBitmapMenuFont
                                     CENTERFLG _ T

```

```

        MENUBORDERSIZE _ 1
        WHENSELECTEDFN _ 'EditBitmapFollowPixelsMatrixDone))
(MatrixWindow (RESETLST
    (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
    (EditBitmapMessage Window "Place the Pixel Follow Matrix Menu.")
    (CREATEW (GETBOXREGION (WIDTHIFWINDOW (fetch IMAGEWIDTH of MatrixMenu)
        4)
        (IPLUS (HEIGHTIFWINDOW (fetch IMAGEHEIGHT of MatrixMenu)
            T 4)))
        "Follow Matrix"))))
(MatrixWindow (ADDMENU MatrixMenu MatrixWindow))
(DoneWindow (ATTACHMENU DoneMenu MatrixWindow 'TOP 'JUSTIFY]
(for Item in (fetch ITEMS of MatrixMenu) do (SHADEITEM Item MatrixMenu BLACKSHADE))
(WINDOWPROP MatrixWindow 'Elements (COPY Positions))
(WINDOWPROP MatrixWindow 'CLOSEFN 'DON'T)
(WINDOWPROP DoneWindow 'CLOSEFN 'DON'T)
MatrixWindow])

```

(EditBitmapFollowPixelsMatrixClose

```
[LAMBDA (Window) (* Gaska "11-Nov-88 09:46")
```

```
(* * Close the pixel follow matrix)
```

```

(LET* [(MenuWindow (CAR (WINDOWPROP Window 'ATTACHEDWINDOWS]
(DETACHWINDOW MenuWindow)
(WINDOWPROP MenuWindow 'MENU NIL)
(WINDOWPROP MenuWindow 'CLOSEFN NIL)
(CLOSEW MenuWindow)
(WINDOWPROP Window 'MENU NIL)
(WINDOWPROP Window 'CLOSEFN NIL)
(WINDOWPROP Window 'Elements NIL)
(CLOSEW Window)])

```

(EditBitmapFollowPixelsMatrixDone

```
[LAMBDA (Item Menu Button) (* Gaska "11-Nov-88 09:47")
```

```
(* * WhenselectedFn for pixel follow matrix finished menu)
```

```

(WINDOWPROP (WINDOWPROP (WFROMMENU Menu)
    'MAINWINDOW)
    'Finished Item])

```

(EditBitmapFollowPixelsMatrixSelect

```
[LAMBDA (Item Menu Button) (* Gaska "11-Nov-88 09:47")
```

```
(* * WhenselectedFn for pixel follow matrix)
```

```

(LET ((Element (CADR Item))
(Window (WFROMMENU Menu)))
(if [AND [NULL (EQUAL Element '(0 . 0]
    (MEMBER Element (WINDOWPROP Window 'Elements]
    then [WINDOWPROP Window 'Elements (REMOVE Element (WINDOWPROP Window 'Elements]
        (SHADEITEM Item Menu WHITESHADE)
    else (if (WINDOWPROP Window 'Elements)
        then (ATTACH Element (WINDOWPROP Window 'Elements))
        else (WINDOWPROP Window 'Elements (LIST Element)))
        (SHADEITEM Item Menu BLACKSHADE)])

```

(EditBitmapFollowPixelsRegion

```
[LAMBDA (Window ConnectFn) (* Gaska "11-Nov-88 09:42")
```

```
(* * Follow connected pixels from the bitmap)
```

```

(LET* [(BoxSize (IMAX (IMIN EditBitmapPixelSelectBoxSize 15)
    5))
(Position (EditBitmapGetPosition Window "Select the starting pixel(s) within the box." (
    EditBitmapCreatePixelSelectionCursor
    BoxSize)))
(PixelList (AND Position (EditBitmapLocatePixelOn (WINDOWPROP Window 'Bitmap)
    (CAR Position)
    (CDR Position)
    BoxSize]
(if PixelList
    then (APPLY* ConnectFn Window '((-1 . 1)
        (0 . 1)
        (1 . 1)
        (-1 . 0)
        (0 . 0)
        (1 . 0)
        (-1 . -1)
        (0 . -1)
        (1 . -1))
    PixelList)

```

```
else (EditBitmapMessage Window "No ON pixels found in selected region."))
```

(EditBitmapFollowPixelsSingle

```
[LAMBDA (Window ConnectFn)
```

```
(* Gaska "11-Nov-88 10:09")
```

```
(* * Remove connected pixels from the bitmap)
```

```
(LET ((DisplayWindow (EditBitmapMagnify Window (WINDOWPROP Window 'Bitmap)
                                                'EditBitmapMagnifySelectButtonEventFn "Position pixel under cursor and click
                                                mouse.")))
  (SPAWN.MOUSE)
  (until (WINDOWPROP Window 'SelectedPixel) do (DISMISS 250))
  (LET [(Position (WINDOWPROP Window 'SelectedPixel)
               (WINDOWPROP Window 'SelectedPixel NIL)
               (WINDOWDELPROP DisplayWindow 'CLOSEFN 'DON'T)
               (CLOSEW DisplayWindow)
               (if (AND (POSITIONP Position)
                        (EQ (BITMAPBIT (WINDOWPROP Window 'Bitmap)
                                     (CAR Position)
                                     (CDR Position))
                            1))
                   then (DISMISS 500)
                   (if (EQ (EditBitmapPopUpMenu '(Specify Default)
                                                  "Follow Matrix?")
                          'Specify)
                       then (LET* [(MatrixWindow (EditBitmapFollowPixelsMatrix Window))
                                   (Elements (until (WINDOWPROP MatrixWindow 'Finished)
                                                    do (DISMISS 250)
                                                    finally (RETURN (AND (EQ (WINDOWPROP MatrixWindow 'Finished)
                                                                              'OK)
                                                                              (WINDOWPROP MatrixWindow 'Elements)
                                                                              (EditBitmapFollowPixelsMatrixClose MatrixWindow))
                                   (if Elements
                                       then (APPLY* ConnectFn Window Elements (LIST Position))
                                       (EditBitmapDisplayBitmap Window)))
                       else (APPLY* ConnectFn Window '((-1 . 1)
                                                         (0 . 1)
                                                         (1 . 1)
                                                         (-1 . 0)
                                                         (0 . 0)
                                                         (1 . 0)
                                                         (-1 . -1)
                                                         (0 . -1)
                                                         (1 . -1))
                                   (LIST Position))
                           (EditBitmapDisplayBitmap Window))
                       else (EditBitmapMessage Window "Selected pixel is not on"])]
```

(EditBitmapRemovePixels

```
[LAMBDA (Window Elements PixelList)
```

```
(* Gaska "4-Oct-88 09:59")
```

```
(* * Remove on pixels from the bitmap)
```

```
(RESETLST
 (RESETSAVE (CURSOR WAITINGCURSOR))
 (EditBitmapMessage Window "This might take some time.")
 (EditBitmapRemoveConnectedPixels (WINDOWPROP Window 'Bitmap)
   Elements PixelList Window)
 (EditBitmapMessage Window "Erasing complete.")
 (EditBitmapDisplayBitmap Window))])
```

(EditBitmapSelectBrushOffset

```
[LAMBDA (Brush)
```

```
(* Gaska "11-Nov-88 12:52")
```

```
(* * Select origin of brush)
```

```
(LET ((Width (BITMAPWIDTH Brush))
      (Height (BITMAPHEIGHT Brush)))
  (SELECTQ (EditBitmapPopUpMenu '(Upper% Left Upper% Right Lower% Right Lower% Left Center)
    "Brush Origin?")
    (Upper% Left (CONS 0 (SUB1 Height)))
    (Upper% Right (CONS (SUB1 Width)
                        (SUB1 Height)))
    (Lower% Right (CONS (SUB1 Width)
                        0))
    (Lower% Left (CONS 0 0))
    (Center (CONS (IQUOTIENT Width 2)
                  (IQUOTIENT Height 2)))
    (CONS (IQUOTIENT Width 2)
          (IQUOTIENT Height 2]))
```

```
)
```

(* * Pattern functions)

(DEFINEQ

(**EditBitmapAddPattern**

[LAMBDA (Window)

(* edited%: "31-Aug-88 10:48")

(* * Add a pattern)

```
(LET* ((Pattern (EditBitmapMakePattern Window)))
  (if Pattern
    then [if [NULL (MEMBER Pattern (WINDOWPROP Window 'Patterns])
              then (if (WINDOWPROP Window 'Patterns)
                        then (NCONC1 (WINDOWPROP Window 'Patterns)
                                     Pattern)
                        else (WINDOWPROP Window 'Patterns (LIST Pattern)
                             Pattern])]
    else (WINDOWPROP Window 'Patterns (LIST Pattern)
         Pattern]))
```

(**EditBitmapCopyEditPattern**

[LAMBDA (Window)

(* Gaska "23-Sep-88 10:09")

(* * Copy and edit a pattern)

```
(if (WINDOWPROP Window 'Patterns)
  then [LET* [(Which (EditBitmapPatternMenu Window T))
              (NewPattern (AND Which (EditBitmap (EditBitmapCopyBitmap Which)
                                         (if NewPattern
                                           then (if (AND (ILEQ (BITMAPWIDTH NewPattern)
                                                                EditBitmapMaxPatternSize)
                                                         (ILEQ (BITMAPHEIGHT NewPattern)
                                                                EditBitmapMaxPatternSize))
                                           then (NCONC1 (WINDOWPROP Window 'Patterns)
                                                         NewPattern)
                                           else (EditBitmapMessage Window (LIST "Pattern is too large." "You may increase the
                                                                                   value of EditBitmapMaxPatternSize."
                                                                                   (CONCAT "The current value is "
                                                                                   EditBitmapMaxPatternSize ".")
                                                                                   EditBitmapMaxPatternSize ".")
                                           else (EditBitmapMessage Window "No patterns available for editing"])]
```

(**EditBitmapDeletePattern**

[LAMBDA (Window)

(* Gaska "15-Sep-88 15:21")

(* * Delete a pattern)

```
(if (WINDOWPROP Window 'Patterns)
  then [LET* [(Which (EditBitmapPatternMenu Window T))
              (if (AND Which (LET* [(PatternWindow (EditBitmapShowPattern Window Which))
                                   (Answer (LET NIL (EditBitmapMessage Window '("Do you really want to
                                                                                   delete this pattern?"
                                                                                   "Click left to
                                                                                   confirm, right to
                                                                                   abort."))
                                   (MOUSECONFIRM "" ""])
                                   (CLOSEW PatternWindow)
                                   (EditBitmapMessageClose Window)
                                   Answer))
              then (if (EQ (LENGTH (WINDOWPROP Window 'Patterns))
                          1)
                        then (WINDOWPROP Window 'Patterns NIL)
                        else (DREMOVE Which (WINDOWPROP Window 'Patterns)
                                         EditBitmapMaxPatternSize))
              else (EditBitmapMessage Window "No patterns available for deletion"])]
```

(**EditBitmapEditPattern**

[LAMBDA (Window)

(* Gaska "23-Sep-88 10:09")

(* * Edit a pattern)

```
(if (WINDOWPROP Window 'Patterns)
  then [LET* [(Which (EditBitmapPatternMenu Window T))
              (NewPattern (AND Which (EditBitmap (EditBitmapCopyBitmap Which)
                                         (if NewPattern
                                           then (if (AND (ILEQ (BITMAPWIDTH NewPattern)
                                                                EditBitmapMaxPatternSize)
                                                         (ILEQ (BITMAPHEIGHT NewPattern)
                                                                EditBitmapMaxPatternSize))
                                           then (RPLACA (MEMBER Which (WINDOWPROP Window 'Patterns)
                                                         NewPattern)
                                           else (EditBitmapMessage Window (LIST "Pattern is too large." "You may increase the
                                                                                   value of EditBitmapMaxPatternSize."
                                                                                   (CONCAT "The current value is "
                                                                                   EditBitmapMaxPatternSize ".")
                                                                                   EditBitmapMaxPatternSize ".")
                                           else (EditBitmapMessage Window "No patterns available for editing"])]
```

(EditBitmapFetchPatterns

[LAMBDA (Window)

(* Gaska "23-Sep-88 09:43")

(* * Fetch patterns from a file)

```

(LET* [(File (EditBitmapGetFileName Window NIL T))
      (Patterns (AND File (LET [(Stream (OPENSTREAM File 'INPUT])
                                (if File
                                    then (RESETLST
                                           (RESETSAVE (CURSOR WAITINGCURSOR))
                                           (RESETSAVE NIL (LIST 'CLOSEF Stream))
                                           (until (EOFP Stream) collect (EditBitmapReadBitmap Stream))))))
      (if Patterns
          then (if (OR (NULL (WINDOWPROP Window 'Patterns))
                      (EQ (EditBitmapPopupMenu ' (REPLACE ADD)
                                                "Options")
                          'REPLACE))
                  then (WINDOWPROP Window 'Patterns Patterns)
                  else (if (WINDOWPROP Window 'Patterns)
                          then (NCONC (WINDOWPROP Window 'Patterns)
                                       Patterns)
                          else (WINDOWPROP Window 'Patterns Patterns)))
          (EditBitmapMessage Window (CONCAT "Patterns retrieved from file: " (FULLNAME File))

```

(EditBitmapFontStylesheet

[LAMBDA (Position InitialValues)

(* Gaska "21-Oct-88 16:33")

(* * Create a font stylesheet)

```

(STYLESHEET (CREATE.STYLE 'TITLE "Font Style" 'ITEM.TITLES ' ("Family" "Size" "Face")
                          'ITEM.TITLE.FONT EditBitmapMenuFont 'ITEMS
                          [LIST (create MENU
                                      ITEMS _ ' (CLASSIC MODERN TERMINAL TITAN GACHA HELVETICA TIMESROMAN))
                                (create MENU
                                      ITEMS _ ' (6 8 10 12 14 18 24 36))
                                (create MENU
                                      ITEMS _ ' (MRR BRR MIR BIR)
                                      'POSITION Position 'SELECTIONS InitialValues])

```

(EditBitmapGetPattern

[LAMBDA (Window)

(* Gaska "1-Sep-88 07:39")

(* * Get a pattern for tessellation or painting)

```

(if (WINDOWPROP Window 'Patterns)
    then (EditBitmapPatternMenu Window)
    else (EditBitmapAddPattern Window))

```

(EditBitmapMakePattern

[LAMBDA (Window)

(* Gaska "25-Oct-88 11:06")

(* * Construct a pattern for tessellation)

```

(LET ((Selection (EditBitmapPopupMenu ' (|Create With Bitmap Editor| |Create With Pixel Editor| From% Screen
                                         |Exact Size From Screen|)
                                     "Pattern Options"))
      (SELECTQ Selection
        (|Create With Bitmap Editor|
          (LET* ((NewPattern (EditBitmap NIL NIL NIL NIL EditBitmapMaxPatternSize
                                     EditBitmapMaxPatternSize)))
            (if NewPattern
                then (if (AND (ILEQ (BITMAPWIDTH NewPattern)
                                   EditBitmapMaxPatternSize)
                              (ILEQ (BITMAPHEIGHT NewPattern)
                                   EditBitmapMaxPatternSize))
                    then NewPattern
                    else (EditBitmapMessage Window (LIST "Pattern is too large." "You may increase
                                                           the value of EditBitmapMaxPatternSize."
                                                           (CONCAT "The current value is "
                                                           EditBitmapMaxPatternSize "."))))
                NIL)))
        (|Create With Pixel Editor|
          [LET* [(Position (CREATEPOSITION LASTMOUSEX LASTMOUSEY))
                (PatternWidth (READNUM "Pattern Width" Position EditBitmapMenuFont
                                     EditBitmapMinPatternSize EditBitmapMaxPatternSize NIL T))
                (PatternHeight (AND PatternWidth (READNUM "Pattern Height" Position EditBitmapMenuFont
                                                         EditBitmapMinPatternSize
                                                         EditBitmapMaxPatternSize NIL T))
                (if PatternHeight
                    then (CAR (NLSETQ (EDITBM (BITMAPCREATE PatternWidth PatternHeight))
                                     (From% Screen (RESETLST
                                                         (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
                                                         (EditBitmapMessage Window "Select an area anywhere on the screen")
                                                         (EditBitmapGetScreenBitmap EditBitmapMinPatternSize EditBitmapMinPatternSize

```

```

                                EditBitmapMaxPatternSize EditBitmapMaxPatternSize)))
(|Exact Size From Screen|
  (LET* [(Position (CREATEPOSITION LASTMOUSEX LASTMOUSEY))
        (PatternWidth (READNUM "Pattern Width" Position EditBitmapMenuFont
                                EditBitmapMinPatternSize EditBitmapMaxPatternSize NIL T))
        (PatternHeight (AND PatternWidth (READNUM "Pattern Height" Position EditBitmapMenuFont
                                EditBitmapMinPatternSize
                                EditBitmapMaxPatternSize NIL T)
        (if PatternHeight
            then (LET* [(Region (GETBOXREGION PatternWidth PatternHeight))
                      (Bitmap (BITMAPCREATE (fetch WIDTH of Region)
                                             (fetch HEIGHT of Region]
                      (BITBLT (SCREENBITMAP)
                              (fetch LEFT of Region)
                              (fetch BOTTOM of Region)
                              Bitmap 0 0 (fetch WIDTH of Region)
                              (fetch HEIGHT of Region))
                      Bitmap)))]
            NIL])

```

(EditBitmapPatternMenu

[LAMBDA (Window NoOther?)

(* Gaska "19-Sep-88 15:45")

(* * Choose an available pattern)

```

(LET* [[Items (APPEND (WINDOWPROP Window 'Patterns)
                      (if (NULL NoOther?)
                          then ' (Other]
  (Selection (MENU (create MENU
                          ITEMS _ Items
                          TITLE _ "Available Patterns"
                          CENTERFLG _ T
                          MENCOLUMNS _ (IQUOTIENT (IPLUS (LENGTH Items)
                                                          4)
                                                    5)
                          MENUBORDERSIZE _ 1
                          MENUFONT _ EditBitmapMenuFont]
  (if Selection
      then (if (EQ Selection 'Other)
                then (EditBitmapAddPattern Window)
                else Selection])

```

(EditBitmapShowPattern

[LAMBDA (Window Bitmap)

(* edited%: "31-Aug-88 10:23")

(* * Show a pattern)

```

(GETMOUSESTATE)
(LET* ((Border 8)
  (Window (CREATEW (CREATEREGION (IPLUS (fetch LEFT of (WINDOWPROP Window 'REGION))
                                         (IQUOTIENT (IDIFFERENCE (WINDOWPROP Window 'WIDTH)
                                                             (IPLUS (BITMAPWIDTH Bitmap)
                                                             Border))
                                         2))
              (IPLUS (fetch BOTTOM of (WINDOWPROP Window 'REGION))
                      (IQUOTIENT (IDIFFERENCE (WINDOWPROP Window 'HEIGHT)
                                              (IPLUS (BITMAPHEIGHT Bitmap)
                                              Border))
                      2))
              (WIDTHIFWINDOW (BITMAPWIDTH Bitmap)
                              Border)
              (HEIGHTIFWINDOW (BITMAPHEIGHT Bitmap)
                              Border))
        NIL Border)))
(BITBLT Bitmap NIL NIL Window)
Window])

```

(EditBitmapStorePatterns

[LAMBDA (Window)

(* Gaska "23-Sep-88 09:44")

(* * Store patterns on a file)

```

(if (WINDOWPROP Window 'Patterns)
    then [LET* ((File (EditBitmapGetFileName Window))
              (if File
                  then (LET [(Stream (OPENSTREAM File 'OUTPUT)
                                (RESETLST
                                  (RESETSAVE (CURSOR WAITINGCURSOR))
                                  (RESETSAVE NIL (LIST 'CLOSEF Stream))
                                  (for Pattern in (WINDOWPROP Window 'Patterns) do (EditBitmapWriteBitmap
                                                                                      Stream Pattern))
                                  (EditBitmapMessage Window (CONCAT "Patterns stored on file: " (FULLNAME
                                                                                      File)))))]
                  else (EditBitmapMessage Window "No patterns available"])]

```


(EditBitmapTrimPattern

[LAMBDA (Window)

; Edited 11-Nov-88 14:51 by Gaska

(* * Edit a pattern)

```

(if (WINDOWPROP Window 'Patterns)
  then (LET* [(Which (EditBitmapPatternMenu Window T))
              (NewPattern (AND Which (RESETLST
                                     (RESETSAVE (CURSOR WAITINGCURSOR))
                                     (EditBitmapTrim Which)))]
            (if NewPattern
                then (RPLACA (MEMBER Which (WINDOWPROP Window 'Patterns))
                             NewPattern))
            else (EditBitmapMessage Window "No patterns available for trimming"])]
)
```

(* * Distort functions)

(DEFINEQ

(EditBitmapBitmapDistort

[LAMBDA (Original Transformed XStart YStart DeltaX DeltaY InitialWidth InitialHeight DeltaWidth DeltaHeight Block) (* Gaska "24-Oct-88 11:08")

(* * Transform the bitmap.)

```

(LET* ((Width (BITMAPWIDTH Original))
      (Height (BITMAPHEIGHT Original))
      (NewWidth (BITMAPWIDTH Transformed))
      (NewHeight (BITMAPHEIGHT Transformed))
      (WidthRatio (FQUOTIENT Width NewWidth))
      (HeightRatio (FQUOTIENT Height NewHeight)))
  (bind (OldX _ 0)
        (YT _ YStart)
        (DH _ InitialHeight)
    do (bind (OldY _ 0)
            (XT _ XStart)
            (DW _ InitialWidth)
        do (if (EQ (BITMAPBIT Original OldX OldY)
                    1)
              then (BITMAPBIT Transformed (FPLUS XT (FTIMES X (FQUOTIENT DW NewWidth)))
              (FPLUS YT (FTIMES Y (FQUOTIENT DH NewHeight)))
              1))
          (SETQ XT (FPLUS XT DeltaX))
          (SETQ DW (FPLUS DW DeltaWidth))
          (SETQ OldY (FPLUS OldY HeightRatio))
          (if Block
              then (BLOCK Block)))
      (SETQ YT (FPLUS YT DeltaY))
      (SETQ DH (FPLUS DH DeltaHeight))
      (SETQ OldX (FPLUS OldX WidthRatio])
```

(EditBitmapDistort

[LAMBDA (Window)

(* Gaska "26-Oct-88 10:09")

(* * Distort a bitmap)

```

(LET ((Vertices (EditBitmapGetQuadrilateral Window)))
  (if Vertices
    then (LET* ((OriginalVertices (CAR Vertices))
                (NewVertices (EditBitmapExpandQuadrilateral (CADR Vertices)))
                (OriginalExtents (EditBitmapGetExtents OriginalVertices 1))
                (NewExtents (EditBitmapGetExtents NewVertices 0)))
          (if NewExtents
              then (LET* [(Block? (EditBitmapBlock?))
                          (OldBitmap (EditBitmapCopyBitmap (WINDOWPROP Window 'Bitmap)
                                                              (CAR OriginalExtents)
                                                              (CADR OriginalExtents)
                                                              (IDIFFERENCE (CADDR OriginalExtents)
                                                              (CAR OriginalExtents))
                                                              (IDIFFERENCE (CADDR OriginalExtents)
                                                              (CADR OriginalExtents))
                                                              (CADDR OriginalExtents))
                          (NewBitmap (BITMAPCREATE (IDIFFERENCE (CADDR NewExtents)
                                                                  (CAR NewExtents))
                                                                  (IDIFFERENCE (CADDR NewExtents)
                                                                  (CADR NewExtents))
                                                                  (CADDR NewExtents))
                          (RESETLST
                            (RESETSAVE (CURSOR (if Block?
                                                       then T
                                                       else WAITINGCURSOR))))
                            (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
                            (if Block?
```

```

    then (SPAWN.MOUSE))
  (EditBitmapMessage Window "This WILL take a LONG time!")
  (EditBitmapDistortBitmap OldBitmap NewBitmap NewVertices Block?)
  (BITBLT NewBitmap 0 0 (WINDOWPROP Window 'Bitmap)
    (IDIFFERENCE (IQUOTIENT (IPLUS (CADDR OriginalExtents)
      (CAR OriginalExtents))
      2)
    (IQUOTIENT (BITMAPWIDTH NewBitmap)
      2))
    (IDIFFERENCE (IQUOTIENT (IPLUS (CADDR OriginalExtents)
      (CADR OriginalExtents))
      2)
    (IQUOTIENT (BITMAPHEIGHT NewBitmap)
      2))
    (BITMAPWIDTH NewBitmap)
    (BITMAPHEIGHT NewBitmap)
    'INPUT
    (WINDOWPROP Window 'Operation))
  (EditBitmapDisplayBitmap Window)))

```

(EditBitmapDistortBitmap

[LAMBDA (Original Transformed NewVertices Block)

(* Gaska "24-Oct-88 10:40")

(* * Distort a bitmap)

```

(LET* ((Extents (EditBitmapGetExtents NewVertices 0))
  (Deltas (EditBitmapGetDeltas NewVertices)))
  (EditBitmapBitmapDistort Original Transformed (IDIFFERENCE (CAAR NewVertices)
    (CAR Extents))
    (IDIFFERENCE (CDAR NewVertices)
      (CADR Extents))
    (CAR Deltas)
    (CADR Deltas)
    (IDIFFERENCE (CAADR NewVertices)
      (CAAR NewVertices))
    (IDIFFERENCE (CDR (CADDR NewVertices))
      (CDAR NewVertices))
    (CADDR Deltas)
    (CADDR Deltas)
    Block])

```

(EditBitmapDrawQuadrilateral

[LAMBDA (WindowOrBitmap Vertices XOffset YOffset)

(* Gaska "11-Oct-88 11:22")

(* * Draw a quadrilateral)

```

(if Vertices
  then (LET* ([Stream (COND
    ((WINDOWP WindowOrBitmap)
      (WINDOWPROP WindowOrBitmap 'DSP))
    ((BITMAPP WindowOrBitmap)
      (DSPCREATE WindowOrBitmap))
    (T (SHOULDNT "Not a window or bitmap"]
      (XOffset (OR XOffset 0))
      (YOffset (OR YOffset 0)))
    (for Start in Vertices as Finish in (CDR Vertices) do (DRAWLINE (IDIFFERENCE (CAR Start)
      XOffset)
      (IDIFFERENCE (CDR Start)
        YOffset)
      (IDIFFERENCE (CAR Finish)
        XOffset)
      (IDIFFERENCE (CDR Finish)
        YOffset)
      1
      'INVERT Stream])

```

(EditBitmapExpandQuadrilateral

[LAMBDA (Vertices)

(* Gaska "13-Oct-88 14:09")

(* * Expand a quadrilateral by a little bit)

```

(LET [[Left (CAR (for Point in Vertices smallest (CAR Point)
  [Bottom (CDR (for Point in Vertices smallest (CDR Point)
  [Right (CAR (for Point in Vertices largest (CAR Point)
  [Top (CDR (for Point in Vertices largest (CDR Point)
  (for Point in Vertices collect (CONS (COND
    ((EQ (CAR Point)
      Right)
      (IPLUS Right 2))
    (T (CAR Point)))
    (COND
      ((EQ (CDR Point)
        Top)
        (IPLUS Top 2))

```

(T (CDR Point])

(EditBitmapGetDeltas

[LAMBDA (NewVertices)

(* Gaska "20-Oct-88 10:53")

(* * Compute delta x, delta y, delta width, and delta height)

```

(LET* ([DeltaX (FQUOTIENT (IDIFFERENCE (CAR (CADDR NewVertices))
                                     (CAAR NewVertices))
                        (IDIFFERENCE (CDR (CADDR NewVertices))
                                     (CDAR NewVertices))
      [DeltaY (FQUOTIENT (IDIFFERENCE (CDADR NewVertices)
                                     (CDAR NewVertices))
                        (IDIFFERENCE (CAADR NewVertices)
                                     (CAAR NewVertices))
      (DeltaWidth (FDIFFERENCE (FQUOTIENT (IDIFFERENCE (CAADDR NewVertices)
                                     (CAADR NewVertices))
                                     (IDIFFERENCE (CDADDR NewVertices)
                                     (CDADR NewVertices)))
                        DeltaX))
      (DeltaHeight (FDIFFERENCE (FQUOTIENT (IDIFFERENCE (CDADDR NewVertices)
                                     (CDR (CADDR NewVertices)))
                                     (IDIFFERENCE (CAADDR NewVertices)
                                     (CAR (CADDR NewVertices)
                                     DeltaY)))
      (LIST DeltaX DeltaY DeltaWidth DeltaHeight])

```

(EditBitmapGetExtents

[LAMBDA (Vertices Margin)

(* Gaska "12-Oct-88 12:16")

(* * Get the minimums and maximums from a set of vertices)

```

(LIST (CAR (for Point in Vertices smallest (CAR Point)))
      (CDR (for Point in Vertices smallest (CDR Point)))
      (IPLUS (CAR (for Point in Vertices largest (CAR Point)))
              Margin)
      (IPLUS (CDR (for Point in Vertices largest (CDR Point)))
              Margin])

```

(EditBitmapGetNewVertices

[LAMBDA (Window OldVertices Grid)

(* Gaska "9-Nov-88 11:02")

(* * Get the new vertices of the quadrilateral)

```

(LET* ((X (EditBitmapGetGriddedValue (LASTMOUSEX Window)
                                     Grid))
      (Y (EditBitmapGetGriddedValue (LASTMOUSEY Window)
                                     Grid)))
      (if (INSIDEP (DSPCLIPPINGREGION NIL (WINDOWPROP Window 'DSP))
                X Y)
          then (LET* [(NewVertices (COPY OldVertices))
                    [Vertex (for Point in OldVertices smallest (IPLUS (EXPT (IABS (IDIFFERENCE (CAR Point)
                                                                                               X))
                                                                                               2)
                                                                                               (EXPT (IABS (IDIFFERENCE (CDR Point)
                                                                                               Y))
                                                                                               2)
                    (Neighbors (for Before in (APPEND (LIST (CADDR OldVertices))
                                                         OldVertices)
                              as It in OldVertices as After in (CDR OldVertices)
                              do (if (EQUAL It Vertex)
                                      then (RETURN (LIST Before After]
                    (DSUBST (CONS X Y)
                              Vertex NewVertices)
          (* * Compute the dot product of the two vectors and if it is negative, the angle is concave and we don't allow that.)
          (if (IGEQ (IDIFFERENCE (ITIMES (IDIFFERENCE X (CAAR Neighbors))
                                     (IDIFFERENCE (CDADR Neighbors)
                                     Y))
                (ITIMES (IDIFFERENCE Y (CDAR Neighbors))
                (IDIFFERENCE (CAADR Neighbors)
                X)))
              0)
              then NewVertices
              else OldVertices])

```

(EditBitmapGetQuadrilateral

[LAMBDA (Window)

(* Gaska "25-Oct-88 13:04")

(* * Fetch the vertices of a quadrilateral)

(RESETLST

(* * Parameter display and setting functions)

(* Gaska "23-Sep-88 09:37")

(* Gaska "23-Sep-88 09:22")

(EditBitmapOperationMenu

```
[LAMBDA (Window InitialSelection)                                (* Gaska "16-Sep-88 15:00")

(** Operation menu)

(LET* ([Menu (create MENU
    ITEMS _ '(PAINT REPLACE INVERT ERASE)
    MENUROWS _ (if (ILESSP (ITIMES (IPLUS (STRINGWIDTH "REPLACE" EditBitmapMenuFont)
                                     4)
                        4)
                    (WINDOWPROP Window 'WIDTH))
        then 1
        else 2)
    MENUFONT _ EditBitmapMenuFont
    MENUBORDERSIZE _ 2
    CENTERFLG _ T
    WHENSELECTEDFN _ (FUNCTION (LAMBDA (Item Menu Button)
                              (if (fetch SHADEDITEMS of Menu)
                                  then (SHADEITEM [CAR (NTH (fetch ITEMS of Menu)
                                                             (CAAR (fetch SHADEDITEMS
                                                                of Menu])
                                                            Menu WHITESHADE)))
                                  (SHADEITEM Item Menu GRAYSHADE)
                                  (WINDOWPROP (WINDOWPROP (WFROMMENU Menu)
                                                           'MAINWINDOW)
                                              'Operation Item]
                              )
      (MenuWindow (MENUWINDOW Menu)))
  [WINDOWPROP MenuWindow 'CLOSEFN ' (FUNCTION (LAMBDA (Window)
                                                (WINDOWPROP Window 'MENU NIL)
      (WINDOWPROP Window 'OperationsMenu Menu)
      (ATTACHWINDOW MenuWindow Window 'TOP 'JUSTIFY)
      (DOSELECTEDITEM Menu InitialSelection)])
```

```

(LET* [(Dashing (MENU (create MENU
                        ITEMS _
                        [APPEND (for DashPattern in EditBitmapListOfDashings
                                collect (LIST (LET* ((Bitmap (BITMAPCREATE 120 20))
                                                       (Stream (DSPCREATE Bitmap)))
                                                       (DRAWLINE 10 9 90 9 1 'REPLACE Stream NIL
                                                       DashPattern)
                                                       (if (EQUAL DashPattern (WINDOWPROP Window
                                                       'Dashing))
                                                           then (BITBLT EditBitmapMenuPointer 0 0 Bitmap
                                                           100 0 20 20))
                                                       Bitmap)
                                                       DashPattern "Selects this dashing pattern"))
                                '((Other Other "Define a new pattern"))
                                (LIST (LET* ((Bitmap (BITMAPCREATE 120 20))
                                               (Stream (DSPCREATE Bitmap)))
                                               (DSPFONT EditBitmapMenuFont Stream)
                                               (CENTERPRINTINREGION "None" (CREATEREGION 0 0 120 20)
                                               Stream)
                                               (if (NULL (WINDOWPROP Window 'Dashing))
                                                   then (BITBLT EditBitmapMenuPointer 0 0 Bitmap 100 0 20 20
                                                   ))
                                               (LIST Bitmap 'None "Remove dashing"]
                                WHENSELECTEDFN _ [FUNCTION (LAMBDA (Item Menu Button)
                                                            (CADR Item)
                                                            MENUBORDERSIZE _ 1
                                                            MENUFONT _ EditBitmapMenuFont
                                                            TITLE _ "DASHING"
                                                            CENTERFLG _ T
                                                            ITEMWIDTH _ 120]
                                (COND
                                [(EQUAL Dashing 'Other)
                                (LET ((NewDashing (EditBitmapGetNewDashing)))
                                (if NewDashing
                                    then (WINDOWPROP Window 'Dashing NewDashing)
                                    (NCONC1 EditBitmapListOfDashings NewDashing]
                                [(EQUAL Dashing 'None)
                                (WINDOWPROP Window 'Dashing NIL))
                                (Dashing (WINDOWPROP Window 'Dashing Dashing))
                                (T NIL)])

```

(EditBitmapSetDrawBrushSize

[LAMBDA (Window)

(* Gaska "2-Sep-88 10:45")

(* Set the brush size for drawing operations for the bitmap.)

```

(LET* ((NewSize (READNUM [LIST "Drawing Brush Size" (CONCAT "Old Value is: " (WINDOWPROP Window 'BrushSize]
                                                                NIL EditBitmapMenuFont 1 EditBitmapMaxBrushSize NIL T)))
      (if NewSize
          then (WINDOWPROP Window 'BrushSize NewSize]))

```

(EditBitmapSetFont

[LAMBDA (Window)

(* Gaska "15-Sep-88 15:22")

(* Set the font style)

```

(LET* [(NewLooks (EditBitmapFontStylesheet (LET NIL (GETMOUSESTATE)
                                                    (CREATEPOSITION LASTMOUSEX LASTMOUSEY))
                                                    (WINDOWPROP Window 'Font]
      (if NewLooks
          then (if (NLSETQ (RESETLST
                            (RESETSAVE (CURSOR WAITINGCURSOR))
                            (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
                            (FONTCREATE NewLooks)
                            T))
                  then (WINDOWPROP Window 'Font NewLooks)
                  else (EditBitmapMessage Window (CONCAT "Can't find " (MKSTRING NewLooks))

```

(EditBitmapSetGrid

[LAMBDA (Window)

(* Gaska "3-Sep-88 11:51")

(* Set the gridding)

```

(WINDOWPROP Window 'Grid (READNUM (LIST (CONCAT "Grid is " (OR (WINDOWPROP Window 'Grid)
                                                                "OFF"))
                                         "Abt turns gridding off")
NIL EditBitmapMenuFont 2 EditBitmapMaxGridSize NIL T))

```

(EditBitmapSetOperation

[LAMBDA (Window)

(* Gaska "16-Sep-88 15:00")

(* Set the default operation for drawing operations on the bitmap.)

```

(LET [(NewOperation (MENU (create MENU
                            ITEMS _ ' (PAINT INVERT REPLACE ERASE)
                            MENUFONT _ EditBitmapMenuFont
                            TITLE _ "Operation"
                            CENTERFLG _ T)))
      (OperationsMenu (WINDOWPROP Window 'OperationsMenu)
      (if NewOperation
          then (DSOPERATION NewOperation Window)
               (WINDOWPROP Window 'Operation NewOperation)
               (SHADEITEM [CAR (NTH (fetch ITEMS of OperationsMenu)
                                   (CAAR (fetch SHADEDITEMS of OperationsMenu)
                                   OperationsMenu WHITESHADE)
               (SHADEITEM NewOperation OperationsMenu GRAYSHADE))

```

(EditBitmapSetOrthogonal

```
[LAMBDA (Window) (* Gaska "15-Sep-88 15:23")
```

```
(* * Toggle the orthogonal lines flag)
```

```

(WINDOWPROP Window 'Orthogonal (NOT (WINDOWPROP Window 'Orthogonal)
(EditBitmapMessage Window (CONCAT "Orthogonal line drawing is now " (if (WINDOWPROP Window 'Orthogonal)
                                                                    then "ON"
                                                                    else "OFF"))

```

(EditBitmapSetPaintBrushSize

```
[LAMBDA (Window) (* Gaska "2-Sep-88 10:44")
```

```
(* * Set the paint brush size)
```

```

(LET* ((NewSize (READNUM [LIST "Paint Brush Size" (CONCAT "Old Value is: " (WINDOWPROP Window
                                                                    'PaintBrushSize]
                                                                    NIL EditBitmapMenuFont 2 16 NIL T)))
      (if NewSize
          then (WINDOWPROP Window 'PaintBrushSize NewSize))

```

(EditBitmapSetRegionGrid

```
[LAMBDA (Window) (* Gaska "23-Sep-88 12:11")
```

```
(* * Set the gridding)
```

```

(WINDOWPROP Window 'RegionGrid (READNUM (LIST (CONCAT "Region Grid is " (OR (WINDOWPROP Window 'RegionGrid)
                                                                    "OFF"))
                                                                    "Abt turns gridding off")
NIL EditBitmapMenuFont 2 EditBitmapMaxGridSize NIL T])

```

(EditBitmapSetShade

```
[LAMBDA (Window) (* Gaska "26-Sep-88 08:16")
```

```
(* * Set the shade)
```

```

(LET* [(Texture (MENU (create MENU
                          ITEMS _
                          [APPEND (for Shade in EditBitmapListOfTextures
                                  collect (LIST (LET ((Bitmap (BITMAPCREATE 120 16)))
                                                  (BITBLT NIL NIL NIL Bitmap NIL NIL 100 NIL
                                                  'TEXTURE
                                                  'REPLACE Shade)
                                                  (if (EQUAL Shade (WINDOWPROP Window 'Shade))
                                                      then (BITBLT EditBitmapMenuPointer 0 0 Bitmap
                                                          100 0 20 16))
                                                  Bitmap)
                                                  Shade "This shade becomes the default texture.))
                                                  ' ("4 X 4 SHADE" NIL "Define a new shade")
                                                  ("16 X 16 SHADE" NIL "Define a new shade")
                          MENUBORDERSIZE _ 1
                          MENUOUTLINESIZE _ 2
                          MENUFONT _ EditBitmapMenuFont
                          TITLE _ "NEW SHADE"
                          CENTERFLG _ T
                          WHENSELECTEDFN _ [FUNCTION (LAMBDA (Item Menu Button)
                                                      (LIST Item Button)
                          ITEMWIDTH _ 120)))
      (NewTexture (COND
                    ((EQUAL (CAAR Texture)
                             "4 X 4 SHADE")
                     (LET ((NewTexture (EDITSHADE)))
                         (NCONC1 EditBitmapListOfTextures NewTexture)
                         NewTexture))
                    ((EQUAL (CAAR Texture)
                             "16 X 16 SHADE")
                     (LET ((NewTexture (EDITSHADE T)))
                         (NCONC1 EditBitmapListOfTextures NewTexture)
                         NewTexture))

```

```

(Texture (CADAR Texture])
(if NewTexture
  then (WINDOWPROP Window 'Shade NewTexture))

```

(EditBitmapShowParameters

[LAMBDA (Window)

(* Gaska "23-Sep-88 11:07")

(* * Display the parameters for drawing on the bitmap.)

```

(EditBitmapMessage Window (LIST [CONCAT "Bitmap Width - " (BITMAPWIDTH (WINDOWPROP Window 'Bitmap]
[CONCAT "Bitmap Height - " (BITMAPHEIGHT (WINDOWPROP Window 'Bitmap]
[CONCAT "Draw Brush Size - " (WINDOWPROP Window 'BrushSize))
[CONCAT "Paint Brush Size - " (WINDOWPROP Window 'PaintBrushSize))
[CONCAT "Air Brush Size - " (WINDOWPROP Window 'AirBrushSize))
[CONCAT "Air Brush Speed - " (WINDOWPROP Window 'AirBrushSpeed))
[CONCAT "Brush Shape - " (WINDOWPROP Window 'BrushShape))
[CONCAT "Operation - " (WINDOWPROP Window 'Operation))
[CONCAT "Dashing - " (OR (WINDOWPROP Window 'Dashing)
"None"))
[CONCAT "Shade - " (WINDOWPROP Window 'Shade))
[CONCAT "Font - " (WINDOWPROP Window 'Font))
[CONCAT "Arrow Width - " (WINDOWPROP Window 'ArrowWidth))
[CONCAT "Arrow Height - " (WINDOWPROP Window 'ArrowHeight))
[CONCAT "Region Grid - " (OR (WINDOWPROP Window 'RegionGrid)
"OFF"))
[CONCAT "Grid - " (OR (WINDOWPROP Window 'Grid)
"OFF"))
[CONCAT "Orthogonal Lines - " (if (WINDOWPROP Window 'Orthogonal)
then "ON"
else "OFF"])]
)

```

(* * Interface to TEdit and SKETCH)

(DEFINEQ

(EditBitmapImageObjButtonEventInFn

[LAMBDA (ImageObj Window)

(* Gaska "7-Oct-88 09:09")

(* * A button has gone down inside the bitmap image object. Give the user the option of editing the bitmap)

```

(LET* [(Object (IMAGEOBJPROP ImageObj 'OBJECTDATUM]
(if (EQ (EditBitmapPopUpMenu '(Edit% Bitmap))
'Edit% Bitmap)
then (LET [(NewBitmap (EditBitmap (fetch (BITMAPOBJ BITMAP) of Object]
(if NewBitmap
then (replace (BITMAPOBJ BITMAP) of Object with NewBitmap)
(IMAGEOBJPROP ImageObj 'CACHED.BITMAP NIL)
'CHANGED)])

```

(EditBitmapImageObjectCreate

[LAMBDA (Bitmap)

(* Gaska "7-Oct-88 08:42")

(* * Create an image object using the bitmap)

```

(IMAGEOBJCREATE Bitmap (IMAGEFNSCREATE (FUNCTION BMOBJ.DISPLAYFN)
(FUNCTION BMOBJ.IMAGEBOXFN)
(FUNCTION BMOBJ.PUTFN)
(FUNCTION BMOBJ.GETFN3)
(FUNCTION BMOBJ.COPYFN)
(FUNCTION EditBitmapImageObjButtonEventInFn))

```

(EditBitmapInsertBitmapIntoSketch

[LAMBDA (Window)

(* Gaska "7-Oct-88 10:43")

(* * Insert the bitmap into a SKETCH window)

```

(LET [(SketchWindow (EditBitmapSelectTargetWindow Window 'Sketch 'INSURE.SKETCH]
(if SketchWindow
then (LET [(ImageObj (BITMAPEDITOBJ (WINDOWPROP Window 'Bitmap)
1 0))]
(replace BUTTONEVENTINFN of (fetch IMAGEOBJFNS of ImageObj) with
EditBitmapImageObjButtonEventInFn
)
(LET* [(Position (CADR (GET.BITMAP.POSITION SketchWindow (WINDOWPROP Window 'Bitmap)
'PAINT "Move the figure into place and press the left
button." 0 0))]
(Element (AND Position (INSIDEP SketchWindow Position)
(SKETCH.CREATE.IMAGE.OBJECT ImageObj Position 1]
(if Element
then (SKETCH.ADD.ELEMENT Element (INSURE.SKETCH SketchWindow)))

```


NIL))

(EditBitmapInsertBitmapIntoTedit

[LAMBDA (Window)

(* Gaska "7-Oct-88 09:55")

(* * Insert the bitmap into a Tedit window)

```

(LET [(TeditWindow (EditBitmapSelectTargetWindow Window 'Tedit 'TEXTSTREAM)
  (if TeditWindow
    then (LET ((ImageObj (BITMAPEDITOBJ (WINDOWPROP Window 'Bitmap)
                                          1 0)))
      (replace BUTTONEVENTINFN of (fetch IMAGEOBJFNS of ImageObj) with
                                   EditBitmapImageObjButtonEventInFn
                                   )
      (TEDIT.INSERT.OBJECT ImageObj (TEXTSTREAM TeditWindow)
        (TEDIT.GETPOINT (TEXTSTREAM TeditWindow)]

```

(EditBitmapSelectTargetWindow

[LAMBDA (Window Type TestFn)

(* Gaska "7-Oct-88 10:44")

(* * Select a target window)

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window (LIST (CONCAT "Left-button in a " Type " window")
                                   "or right-button to abort.)))
  (bind Target until (OR Target (LASTMOUSESTATE RIGHT))
    do (UNTILMOUSESTATE (OR LEFT RIGHT))
      (if (LASTMOUSESTATE LEFT)
        then (SETQ Target (WHICHW))
        (if [NULL (CAR (NLSETQ (APPLY* TestFn Target)
                               then (SETQ Target NIL)
                               (EditBitmapMessage Window (LIST (CONCAT "Not a " Type " window.")
                                                                (CONCAT "Left-button in a " Type " window")
                                                                "or right-button to abort.)))
          (UNTILMOUSESTATE UP)))
      finally (UNTILMOUSESTATE UP)
        (RETURN Target)))]

```

)

(* * Very basic input/output functions)

(DEFINEQ

(EditBitmapAirbrushOutline

[LAMBDA (Size Shape)

(* Gaska "28-Oct-88 14:25")

(* * Make outline bitmap for the airbrush)

```

(SETQ Size (IPLUS Size (if (EVENP Size)
                           then 3
                           else 2)))
(LET* ((Bitmap (BITMAPCREATE Size Size))
  (Stream (DSPCREATE Bitmap))
  (HalfSize (IQUOTIENT Size 2)))
  (SELECTQ Shape
    (SQUARE (MOVETO 0 0 Stream)
      (DRAWTO (SUB1 Size)
        0 1 'PAINT Stream)
      (DRAWTO (SUB1 Size)
        (SUB1 Size)
        1
        'PAINT Stream)
      (DRAWTO 0 (SUB1 Size)
        1
        'PAINT Stream)
      (DRAWTO 0 0 1 'PAINT Stream))
    (ROUND (DRAWCIRCLE HalfSize HalfSize HalfSize 1 NIL Stream))
    (HORIZONTAL (MOVETO 0 HalfSize Stream)
      (DRAWTO Size HalfSize 1 'PAINT Stream))
    (VERTICAL (MOVETO HalfSize 0 Stream)
      (DRAWTO HalfSize Size 1 'PAINT Stream))
    (DIAGONAL (MOVETO 0 0 Stream)
      (DRAWTO Size Size 1 'PAINT Stream))
    NIL)
  Bitmap])

```

(EditBitmapAirbrushPaint

[LAMBDA (Window Bitmap BrushShape BrushSize BitsOn Operation Speed TimerIntervals Font)

(* Gaska "28-Oct-88 15:10")

[illegible]


```
(if Block
  then (BLOCK Block])
```

```
NewBitmap])
```

(EditBitmapBitmapInvertVertical

```
[LAMBDA (Bitmap Block)
```

```
(* Gaska "22-Sep-88 09:39")
```

```
(* * Invert a bitmap vertically)
```

```
(LET* ((Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (NewBitmap (EditBitmapCopyBitmap Bitmap)))
  [for X1 from 0 to (SUB1 (IQUOTIENT Height 2)) do (for Y from 0 to (SUB1 Width)
    bind (X2 _ (IDIFFERENCE (SUB1 Height)
                             X1))
    do (BITMAPBIT NewBitmap Y X1 (BITMAPBIT Bitmap Y X2
      ))
      (BITMAPBIT NewBitmap Y X2 (BITMAPBIT Bitmap Y X1
      ))
      (if Block
        then (BLOCK Block])

NewBitmap])
```

(EditBitmapBitmapRotate

```
[LAMBDA (Bitmap Direction Block)
```

```
(* Gaska "21-Sep-88 10:14")
```

```
(* * Rotate a bitmap)
```

```
(LET* ((Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (NewBitmap (BITMAPCREATE Height Width)))
  (SELECTQ Direction
    (Left [for Y from 0 to (SUB1 Height) do (for X from 0 to (SUB1 Width)
      bind (Y1 _ (IDIFFERENCE (SUB1 Height)
                              Y))
      do (BITMAPBIT NewBitmap Y1 X (BITMAPBIT Bitmap X Y))
      (if Block
        then (BLOCK Block])

      NewBitmap)
    (Right [for X from 0 to (SUB1 Width) do (for Y from 0 to (SUB1 Height)
      bind (X1 _ (IDIFFERENCE (SUB1 Width)
                              X))
      do (BITMAPBIT NewBitmap Y X1 (BITMAPBIT Bitmap X Y))
      (if Block
        then (BLOCK Block])

      NewBitmap)
    NIL))
```

(EditBitmapBitmapRotateArbitrary

```
[LAMBDA (Bitmap Angle Block)
```

```
(* Gaska "21-Sep-88 10:28")
```

```
(* * Rotate a bitmap by an arbitrary angle)
```

```
(LET* [(Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (CosAngle (COS Angle))
      (SinAngle (SIN Angle))
      [Corners (for Point in (LIST (CONS 0 0)
        (CONS Width 0)
        (CONS Width Height)
        (CONS 0 Height))
        collect (CONS (FDIFFERENCE (FTIMES (CAR Point)
          CosAngle)
          (FTIMES (CDR Point)
            SinAngle))
          (FPLUS (FTIMES (CAR Point)
            SinAngle)
          (FTIMES (CDR Point)
            CosAngle)
          [XMin (CAR (for Point in Corners smallest (CAR Point)
          [XMax (CAR (for Point in Corners largest (CAR Point)
          [YMin (CDR (for Point in Corners smallest (CDR Point)
          [YMax (CDR (for Point in Corners largest (CDR Point)
          (XOffset (FPLUS (ABS XMin)
            1.0))
          (YOffset (FPLUS (ABS YMin)
            1.0))
          (NewBitmap (BITMAPCREATE (IPLUS (EditBitmapRound (ABS (FDIFFERENCE XMax XMin)))
            2)
            (IPLUS (EditBitmapRound (ABS (FDIFFERENCE YMax YMin)))
            2]
          [for Y from 0 to (SUB1 Height)
            do (for X from 0 to (SUB1 Width)
              do (if (EQ (BITMAPBIT Bitmap X Y)
```

```

1)
then (BITMAPBIT NewBitmap [EditBitmapRound (FPLUS XOffset (FTIMES X CosAngle)
                                           (FMINUS (FTIMES Y SinAngle]
                                           (EditBitmapRound (FPLUS YOffset (FTIMES X SinAngle)
                                           (FTIMES Y CosAngle)))
1))
(if Block
  then (BLOCK Block]
NewBitmap])

```

(EditBitmapBitmapShift

```
[LAMBDA (Bitmap XShift YShift) (* Gaska "26-Sep-88 08:34")
```

```
(* * Shift a bitmap)
```

```

(LET* [(Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (NewBitmap (BITMAPCREATE (IPLUS Width (IABS XShift))
                               (IPLUS Height (IABS YShift]
      (BITBLT Bitmap 0 0 NewBitmap (IMAX XShift 0)
              (IMAX YShift 0)
              Width Height)
      NewBitmap])

```

(EditBitmapBorder

```
[LAMBDA (Bitmap Bits Texture) (* Gaska "26-Sep-88 08:34")
```

```
(* * Add a border to a bitmap)
```

```

(LET* [(Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (NewBitmap (BITMAPCREATE (IPLUS Width (ITIMES Bits 2))
                               (IPLUS Height (ITIMES Bits 2]
      (BITBLT NIL NIL NIL NewBitmap NIL NIL NIL 'TEXTURE 'REPLACE (OR Texture WHITESHADE))
      (BITBLT Bitmap 0 0 NewBitmap Bits Bits Width Height)
      NewBitmap])

```

(EditBitmapBoxInput

```
[LAMBDA (Window RegionGrid MinWdith MinHeight) (* Gaska "23-Sep-88 11:14")
```

```
(* * Fetch vertices for a box)
```

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Specify a region within the window")
  (LET ((Region (EditBitmapGetWindowRegion Window RegionGrid MinWdith MinHeight)))
    (if Region
      then (EditBitmapVerticesFromRegion Region))))))

```

(EditBitmapCircleGetRadiusPoint

```
[LAMBDA (Window Grid Center Cursor) (* Gaska " 7-Sep-88 14:44")
```

```
(* * Reads a point from the user prompting them with a circle that follows the cursor)
```

```
(CAR (EditBitmapGetPointAnywhere Window Grid NIL Cursor 'EditBitmapCircleShow Center])
```

(EditBitmapCircleInput

```
[LAMBDA (Window Grid) (* Gaska " 7-Sep-88 14:45")
```

```
(* * Reads two points from the user and returns a circle element that it represents.)
```

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Indicate center of circle")
  (LET ((Center (CAR (EditBitmapGetPoint Window Grid NIL EditBitmapCircleCenter)))
        RadiusPoint)
    (if Center
      then (EditBitmapMarkSpot Center NIL Window)
            (EditBitmapMessage Window "Indicate a point of the circumference of the circle")
            (SETQ RadiusPoint (EditBitmapCircleGetRadiusPoint Window Grid Center EditBitmapCircleEdge)
            )
            (EditBitmapMarkSpot Center NIL Window)
            (LIST Center RadiusPoint))))))

```

(EditBitmapCircleShow

```
[LAMBDA (Window X Y Center) (* Gaska "20-Aug-88 10:05")
```

```
(* * Xors a circle to X Y from Center in a window. Used as the feedback function for reading the radius point for circles.)
```

```

(EditBitmapShowPoint Window X Y)
(DRAWCIRCLE (CAR Center)

```

```
(SELECTQ Field
  (BITMAP (if NewValue
    then (replace CUIIMAGE of Cursor with NewValue)
    else (fetch CUIIMAGE of Cursor)))
  (HOTSPOT (if NewValue
    then (replace CUHOTSPOTX of Cursor with (CAR NewValue))
        (replace CUHOTSPOTY of Cursor with (CDR NewValue))
    else (CREATEPOSITION (fetch CUHOTSPOTX of Cursor)
        (fetch CUHOTSPOTY of Cursor)))))
```

NIL])

(EditBitmapCurveInput

[LAMBDA (Window Grid)

(* Gaska "3-Sep-88 11:18")

(* * Fetch a series of points for a curve)

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window '("Enter the points the curve goes through." "Middle button erases last fixed
                                point." "Left click outside the window to stop.))
  (LET ((Wire (EditBitmapGetPointList Window Grid)))
    (if (AND Wire (IGEQ (LENGTH Wire)
                        2))
        then Wire))))

```

(EditBitmapDistance

[LAMBDA (Arg1 Arg2 Arg3 Arg4)

(* Gaska "12-Aug-88 13:10")

(* * Returns the distance between two points)

```

(LET [(DeltaX (if (POSITIONP Arg1)
                  then (DIFFERENCE (CAR Arg2)
                                    (CAR Arg1))
                  else (IDIFFERENCE Arg3 Arg1)))
      (DeltaY (if (POSITIONP Arg1)
                  then (DIFFERENCE (CDR Arg2)
                                    (CDR Arg1))
                  else (IDIFFERENCE Arg4 Arg2))]
  (SQRT (PLUS (TIMES DeltaX DeltaX)
              (TIMES DeltaY DeltaY)))

```

(EditBitmapDrawOverConnectedPixels

[LAMBDA (Bitmap Elements PixelList FeedbackWindow Brush XOffset YOffset Operation)

(* Gaska "11-Nov-88 12:51")

(* * Draw over connected pixels in a bitmap)

```

(LET* ((TempBitmap (EditBitmapCopyBitmap Bitmap))
      (Stack PixelList))
  (while Stack
    do (LET ((Top (pop Stack)))
      (if (EQ (BITMAPBIT TempBitmap (CAR Top))
              (CDR Top))
          1)
      then (BITMAPBIT TempBitmap (CAR Top)
                      (CDR Top)
                      0)
          (BITBLT Brush 0 0 Bitmap (IDIFFERENCE (CAR Top)
                                                  XOffset)
                  (IDIFFERENCE (CDR Top)
                              YOffset)
                  NIL NIL 'INPUT Operation)
          (if FeedbackWindow
              then (BITBLT Brush 0 0 FeedbackWindow (IDIFFERENCE (CAR Top)
                                                                    XOffset)
                  (IDIFFERENCE (CDR Top)
                              YOffset)
                  NIL NIL 'INPUT Operation))
          (for Position in Elements
            do (LET [(X (IPLUS (CAR Top)
                              (CAR Position)))
                    (Y (IPLUS (CDR Top)
                              (CDR Position)))]
              (if (EQ (BITMAPBIT TempBitmap X Y)
                      1)
                  then (push Stack (CONS X Y))

```

(EditBitmapEditRegion

[LAMBDA (Window Bitmap Editor RegionGrid Grid MinWidth MinHeight)

(* Gaska "26-Sep-88 08:35")

(* * Edit a region)

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Select the region to be edited.")
  (LET* [(Region (EditBitmapGetWindowRegion Window RegionGrid MinWidth MinHeight))
        (NewBitmap (if Region
                        then (BITMAPCREATE (fetch WIDTH of Region)
                                           (fetch HEIGHT of Region))
                        (if Region
                            then (BITBLT Bitmap (fetch LEFT of Region)

```

```

        (fetch BOTTOM of Region)
        NewBitmap 0 0 (fetch WIDTH of Region)
        (fetch HEIGHT of Region))
[SETQ NewBitmap (CAR (NLSETQ (APPLY* Editor NewBitmap)
(if NewBitmap
    then (DSPFILL Region WHITESHADE 'REPLACE (DSPCREATE Bitmap))
        (BITBLT NewBitmap 0 0 Bitmap (fetch LEFT of Region)
            (fetch BOTTOM of Region)
            (BITMAPWIDTH Bitmap)
            (BITMAPHEIGHT Bitmap))
        T)))]])

```

(EditBitmapEllipseGetMajorPoint

[LAMBDA (Window Grid Center)

(* Gaska "7-Sep-88 14:46")

(* * Reads a position from the user that will be the major radius point of an ellipse.)

(CAR (EditBitmapGetPointAnywhere Window Grid NIL EditBitmapEllipseMajor 'EditBitmapEllipseShowMajor Center))

(EditBitmapEllipseGetMinorPoint

[LAMBDA (Window Grid Center MajorPoint)

(* Gaska "7-Sep-88 14:48")

(* * Reads a position from the user that will be the major radius point of an ellipse.)

```

(CAR (EditBitmapGetPointAnywhere Window Grid NIL EditBitmapEllipseMinor 'EditBitmapEllipseShowMinor
      (LIST Center (EditBitmapDistance Center MajorPoint)
              (EditBitmapEllipseOrientation Center MajorPoint))

```

(EditBitmapEllipseInput

[LAMBDA (Window Grid)

(* Gaska "7-Sep-88 14:47")

(* * Reads three points from the user and returns the ellipse figure element that it represents.)

```

(RESETLST
 (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
 (EditBitmapMessage Window "Indicate center of ellipse")
 (LET* (Center MajorRadius MinorRadius)
   (SETQ Center (CAR (EditBitmapGetPoint Window Grid NIL EditBitmapEllipseCenter)))
   (EditBitmapMarkSpot Center NIL Window)
   (if Center
     then (EditBitmapMessage Window "Indicate semi-major axis")
           (SETQ MajorRadius (EditBitmapEllipseGetMajorPoint Window Grid Center))
           (EditBitmapMarkSpot MajorRadius NIL Window)
           (EditBitmapMessage Window "Indicate semi-minor axis")
           (SETQ MinorRadius (EditBitmapEllipseGetMinorPoint Window Grid Center MajorRadius))
           (EditBitmapMarkSpot MajorRadius NIL Window)
           (EditBitmapMarkSpot Center NIL Window)
           (LIST Center MajorRadius MinorRadius))))))

```

(EditBitmapEllipseOrientation

[LAMBDA (Center MajorRadiusPoint)

(* Gaska "12-Aug-88 14:55")

(* * Computes the orientation of an ellipse from its center point and its major radius point.)

```

(LET [(Deltax (IDIFFERENCE (CAR MajorRadiusPoint)
                           (CAR Center))
      (CDR Center))
      (if (ZEROP Deltax)
          then 90.0
          else (ARCTAN2 (IDIFFERENCE (CDR MajorRadiusPoint)
                                      (CDR Center))
                        Deltax))]

```

(EditBitmapEllipseShowMajor

[LAMBDA (Window X Y Center)

(* Gaska "22-Aug-88 13:08")

(* * Used as the feedback function for reading the major radius point for ellipses.)

```

(EditBitmapShowPoint Window X Y)
(DRAWLINE X Y (PLUS X (TIMES 2 (DIFFERENCE (CAR Center)
                                             X)))
          (PLUS Y (TIMES 2 (DIFFERENCE (CDR Center)
                                             Y))))
1
'INVERT Window])

```

(EditBitmapEllipseShowMinor

[LAMBDA (Window X Y Ellipse)

(* Gaska "22-Aug-88 13:10")

(* * Used as the feedback function for reading the minor radius point for ellipses.)

(EditBitmapShowPoint Window X Y)


```
(DRAWELLIPSE (CAAR Ellipse)
  (CDAR Ellipse)
  (EditBitmapDistance (CAAR Ellipse)
    (CDAR Ellipse)
    X Y)
  (CADR Ellipse)
  (CADDR Ellipse)
  1 NIL Window])
```

(EditBitmapFromScreen

```
[LAMBDA (MinWidth MinHeight MaxWidth MaxHeight) (* Gaska "30-Aug-88 16:19")
```

```
(* * Create a bitmap from the screen)
```

```
(PRINTOUT PROMPTWINDOW T "Indicate a region on the screen.")
(EditBitmapGetScreenBitmap MinWidth MinHeight MaxWidth MaxHeight])
```

(EditBitmapGetFileName

```
[LAMBDA (Window Message Input? Candidate) (* Gaska "27-Oct-88 13:02")
```

```
(* * Fetch a file name)
```

```
(LET* ((File (EditBitmapGetStringFromUser Window (OR Message "File Name: ")
  2 Candidate)))
  (if (AND File Input?)
    then (if (FINDFILE File)
      then File
      else (EditBitmapMessage Window (CONCAT "Can't find file: " File))
      NIL)
    else File])
```

(EditBitmapGetGriddedValue

```
[LAMBDA (Value Grid) (* Gaska "23-Sep-88 10:47")
```

```
(* * Get value snapped to grid)
```

```
(if Grid
  then (ITIMES (IQUOTIENT (IPLUS (LLSH Value 1)
    (if (MINUSP Value)
      then (IMINUS Grid)
      else Grid))
    (LLSH Grid 1))
    Grid)
  else Value])
```

(EditBitmapGetPoint

```
[LAMBDA (Window Grid Orthogonal? Cursor FeedBackFn PointList) (* Gaska "3-Oct-88 10:43")
```

```
(* * Gets a point from the user and gives feedback)
```

```
(DECLARE%: (GLOBALVARS CROSSHAIRS))
(UNTILMOUSESTATE UP)
(RESETLST
  (RESETSAVE (CURSOR (OR Cursor CROSSHAIRS)))
  (RESETSAVE NIL (LIST 'DSOPERATION (DSOPERATION 'INVERT Window)
    Window)))
[LET ((FeedBackFn (OR FeedBackFn 'EditBitmapShowPoint))
  (DisplayStream (GETSTREAM Window))
  (TwiceGrid (if Grid
    then (LLSH Grid 1)))
  (X NIL)
  (Y NIL))
  (TOTOPW Window)
  (until (MOUSESTATE (NOT UP))
    do (GETMOUSESTATE)
      (LET ((NewX (EditBitmapGetGriddedValue (LASTMOUSEX DisplayStream)
        Grid))
        (NewY (EditBitmapGetGriddedValue (LASTMOUSEY DisplayStream)
        Grid)))
        (if (OR (NEQ NewX X)
          (NEQ NewY Y))
          then [if (AND Orthogonal? (LISTP PointList))
            then (if [ILEQ [IABS (IDIFFERENCE NewX (CAAR (LAST PointList)
              (IABS (IDIFFERENCE NewY (CDAR (LAST PointList)
                then (SETQ NewX (CAAR (LAST PointList)))
                else (SETQ NewY (CDAR (LAST PointList)
              (if (AND X Y (EditBitmapInsideWindow Window X Y))
                then (APPLY* FeedBackFn Window X Y PointList))
              (if (EditBitmapInsideWindow Window NewX NewY)
                then (APPLY* FeedBackFn Window NewX NewY PointList))
              (SETQ X NewX)
              (SETQ Y NewY))]
            finally (if (EditBitmapInsideWindow Window X Y)
```

```

    then (APPLY* FeedBackFn Window X Y PointList)
      [RETURN (LIST (CREATEPOSITION X Y)
                    (if (LASTMOUSESTATE MIDDLE)
                        then 'ERASE
                        else 'DRAW]
    else (RETURN NIL)])])

```

(EditBitmapGetPointAnywhere

```
[LAMBDA (Window Grid Orthogonal? Cursor FeedBackFn PointList) (* Gaska " 7-Sep-88 14:44")
```

```
(* * Gets a point from the user and gives feedback)
```

```

(DECLARE%: (GLOBALVARS CROSSHAIRS))
(UNTILMOUSESTATE UP)
(RESETLST
  (RESETSAVE (CURSOR (OR Cursor CROSSHAIRS)))
  (RESETSAVE NIL (LIST 'DSOPERATION (DSOPERATION 'INVERT Window)
                      Window))
[LET ((FeedBackFn (OR FeedBackFn 'EditBitmapShowPoint))
      (DisplayStream (GETSTREAM Window))
      (TwiceGrid (if Grid
                      then (LLSH Grid 1))))
  (X NIL)
  (Y NIL)
  (TOTOPW Window)
  (until (MOUSESTATE (NOT UP))
    do (GETMOUSESTATE)
      (LET ((NewX (EditBitmapGetGriddedValue (LASTMOUSEX DisplayStream)
                                             Grid))
            (NewY (EditBitmapGetGriddedValue (LASTMOUSEY DisplayStream)
                                             Grid)))
        (if (OR (NEQ NewX X)
                (NEQ NewY Y))
          then [if (AND Orthogonal? (LISTP PointList))
                  then (if [ILEQ [IABS (IDIFFERENCE NewX (CAAR (LAST PointList)
                                                         (IABS (IDIFFERENCE NewY (CDAR (LAST PointList)
                                                         then (SETQ NewX (CAAR (LAST PointList)))
                  else (SETQ NewY (CDAR (LAST PointList)
                  (if (AND X Y)
                    then (APPLY* FeedBackFn Window X Y PointList))
                    (APPLY* FeedBackFn Window NewX NewY PointList)
                    (SETQ X NewX)
                    (SETQ Y NewY))]
          finally (APPLY* FeedBackFn Window X Y PointList)
                  (RETURN (LIST (CREATEPOSITION X Y)
                                (if (LASTMOUSESTATE MIDDLE)
                                    then 'ERASE
                                    else 'DRAW]]))

```

(EditBitmapGetPointList

```
[LAMBDA (Window Grid Orthogonal? FeedbackFn) (* Gaska " 3-Sep-88 10:53")
```

```
(* * Reads a series of points from the user.)
```

```

(LET (Points)
[bind Value Point repeatuntil (NULL (SETQ Value (EditBitmapGetPoint Window Grid Orthogonal? NIL FeedbackFn
                                                                    Points)))
  do (if Value
      then (SETQ Point (CAR Value))
      (if (EQ (CADR Value)
              'ERASE)
        then [if (IGEQ (LENGTH Points)
                        2)
                then (LET ((TempPoints (REVERSE Points)))
                        (EditBitmapShowPoint Window (CAAR TempPoints)
                                              (CDAR TempPoints))
                        (if FeedbackFn
                          then (DRAWLINE (CAAR TempPoints)
                                           (CDAR TempPoints)
                                           (CAADR TempPoints)
                                           (CDADR TempPoints)
                                           1
                                           'INVERT Window))
                        (SETQ Points (REVERSE (CDR TempPoints))
                        else (EditBitmapShowPoint Window (CAR Point)
                                                          (CDR Point))
                        (if (AND Points FeedbackFn)
                          then (DRAWLINE (CAAR (LAST Points))
                                           (CDAR (LAST Points))
                                           (CAR Point)
                                           (CDR Point)
                                           1
                                           'INVERT Window))
                        (SETQ Points (NCONC1 Points Point]
    (for PointTail on Points do (EditBitmapShowPoint Window (CAAR PointTail)

```

```

(CDAR PointTail))
(AND (CDR PointTail)
FeedbackFn
(DRAWLINE (CAAR PointTail)
(CDAR PointTail)
(CAADR PointTail)
(CDADR PointTail)
1
'INVERT Window)))

Points])

```

(EditBitmapGetPosition

```
[LAMBDA (Window Prompt Cursor) (* Gaska "28-Sep-88 09:08")
```

```
(* * Get a position inside a window)
```

```
(DECLARE (GLOBALVARS CROSSHAIRS))
```

```
(LET [(Position (if Prompt
then (RESETLST
(RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
(EditBitmapMessage Window Prompt)
(GETPOSITION Window (OR Cursor CROSSHAIRS)))
else (GETPOSITION Window (OR Cursor CROSSHAIRS]
(AND Position (INSIDEP Window Position)
Position])
```

(EditBitmapGetScreenBitmap

```
[LAMBDA (MinWidth MinHeight MaxWidth MaxHeight) (* Gaska "26-Sep-88 08:35")
```

```
(* * Fetch a bitmap from the screen)
```

```
(LET* [(Region (EditBitmapGetScreenRegion MinWidth MinHeight MaxWidth MaxHeight))
(Bitmap (BITMAPCREATE (fetch WIDTH of Region)
(fetch HEIGHT of Region]
(BITBLT (SCREENBITMAP)
(fetch LEFT of Region)
(fetch BOTTOM of Region)
Bitmap 0 0 (fetch WIDTH of Region)
(fetch HEIGHT of Region))
Bitmap])
```

(EditBitmapGetScreenRegion

```
[LAMBDA (MinWidth MinHeight MaxWidth MaxHeight) (* Gaska "9-Sep-88 09:27")
```

```
(* * Fetch a region from the screen)
```

```
(GETREGION MinWidth MinHeight NIL 'EditBitmapConstrainRegion (CONS MaxWidth MaxHeight])
```

(EditBitmapGetStringFromUser

```
[LAMBDA (Window Message Lines Candidate) (* Gaska "27-Oct-88 13:01")
```

```
(* * Fetch a string from the user)
```

```
(RESETLST
(RESETSAVE NIL (LIST 'REMOVEPROMPTWINDOW Window))
(PROMPTFORWARD Message Candidate NIL (LET ((PromptWindow (GETPROMPTWINDOW Window (OR Lines 2)
EditBitmapMessageFont)))
(MOVETOUPPERLEFT PromptWindow)
PromptWindow)
NIL
'TTY)))
```

(EditBitmapGetWindowRegion

```
[LAMBDA (Window Grid MinWidth MinHeight) (* Gaska "23-Sep-88 11:20")
```

```
(* * Get a region within a window by first specifying a corner)
```

```
(LET* ([DisplayRegion (CREATEREGION (DSPXOFFSET NIL Window)
(DSPYOFFSET NIL Window)
(WINDOWPROP Window 'WIDTH)
(WINDOWPROP Window 'HEIGHT]
(Region (INTERSECTREGIONS (if Grid
then (GETREGION MinWidth MinHeight NIL 'EditBitmapInsureGriddedRegion
(CONS DisplayRegion Grid))
else (GETREGION MinWidth MinHeight))
DisplayRegion)))
(if (AND Region (IGREATERP (fetch WIDTH of Region)
MinWidth)
(IGREATERP (fetch HEIGHT of Region)
MinHeight))
then (CREATEREGION (IDIFFERENCE (fetch LEFT of Region)
(fetch LEFT of DisplayRegion))
```

```

(IDIFFERENCE (fetch BOTTOM of Region)
  (fetch BOTTOM of DisplayRegion))
(fetch WIDTH of Region)
(fetch HEIGHT of Region))

```

(EditBitmapInsideWindow

[LAMBDA (Window X Y)

(* Gaska "13-Aug-88 09:37")

```

(* * Determine if X, Y is inside a window. X, Y are in the window's display stream coordinates)

```

```

(AND (IGEQ X 0)
  (ILEQ X (WINDOWPROP Window 'WIDTH))
  (IGEQ Y 0)
  (ILEQ Y (WINDOWPROP Window 'HEIGHT)))

```

(EditBitmapInsureGriddedRegion

[LAMBDA (Fixed Moving Constraints)

(* Gaska "23-Sep-88 10:49")

```

(* * Insure that the defined region is within a window)

```

```

(LET ((Region (CAR Constraints))
  (Grid (CDR Constraints)))
  (if Moving
    then (CREATEPOSITION (IPLUS (fetch LEFT of Region)
      (EditBitmapGetGriddedValue (IDIFFERENCE (CAR Moving)
        (fetch LEFT of Region))
        Grid))
      (IPLUS (fetch BOTTOM of Region)
        (EditBitmapGetGriddedValue (IDIFFERENCE (CDR Moving)
          (fetch BOTTOM of Region))
          Grid)))
    else (CREATEPOSITION (IPLUS (fetch LEFT of Region)
      (EditBitmapGetGriddedValue (IDIFFERENCE (CAR Fixed)
        (fetch LEFT of Region))
        Grid))
      (IPLUS (fetch BOTTOM of Region)
        (EditBitmapGetGriddedValue (IDIFFERENCE (CDR Fixed)
          (fetch BOTTOM of Region))
          Grid))
      Grid))

```

(EditBitmapLocatePixelOn

[LAMBDA (Bitmap XStart YStart BoxSize)

(* Gaska "29-Sep-88 07:41")

```

(* * Locate the on pixels within BoxSize with center at (XStart, YStart))

```

```

(LET ((PixelList (CONS)
  (HalfSize (IQUOTIENT BoxSize 2)))
  [for X from (IMINUS HalfSize) to HalfSize
    do (for Y from (IMINUS HalfSize) to HalfSize do (if (EQ (BITMAPBIT Bitmap (IPLUS XStart X)
      (IPLUS YStart Y))
        1)
      then (TCONC PixelList (CONS (IPLUS XStart X)
        (IPLUS YStart Y))
        (CAR PixelList]))

```

(EditBitmapMakeGrid

[LAMBDA (Bitmap Region GridWidth GridHeight LineWidth Operation Dashing Clipped?)

(* Gaska "8-Sep-88 09:56")

```

(* * Display a grid)

```

```

(LET* [(OriginX (fetch LEFT of Region))
  (OriginY (fetch BOTTOM of Region))
  (Offset (if (ODDP LineWidth)
    then (IQUOTIENT LineWidth 2)
    else (SUB1 (IQUOTIENT LineWidth 2)
      (Width (IPLUS (fetch WIDTH of Region)
        LineWidth))
      (Height (IPLUS (fetch HEIGHT of Region)
        LineWidth))
      (TempBitmap (BITMAPCREATE Width Height))
      (Stream (DSPCREATE TempBitmap))
      (HorizontalLines (IPLUS (IQUOTIENT (IDIFFERENCE (fetch HEIGHT of Region)
        LineWidth)
        GridHeight)
      (OR (AND Clipped? 1)
        2)))
      (VerticalLines (IPLUS (IQUOTIENT (IDIFFERENCE (fetch WIDTH of Region)
        LineWidth)
        GridWidth)
      (OR (AND Clipped? 1)
        2]
    (bind (EndOfLine _ (IPLUS Offset (ITIMES (SUB1 VerticalLines)

```

```

                                GridWidth)))
  for Y from Offset to (IPLUS Offset (SUB1 (ITIMES HorizontalLines GridHeight))) by GridHeight
  do (DRAWLINE Offset Y EndOfLine Y LineWidth 'PAINT Stream NIL Dashing))
(bind (EndOfLine _ (IPLUS Offset (ITIMES (SUB1 HorizontalLines)
                                GridHeight)))
  for X from Offset to (IPLUS Offset (SUB1 (ITIMES VerticalLines GridWidth))) by GridWidth
  do (DRAWLINE X Offset X EndOfLine LineWidth 'PAINT Stream NIL Dashing))
(BITBLT TempBitmap 0 0 Bitmap OriginX OriginY NIL NIL 'INPUT Operation])

```

(EditBitmapMakeMask

[LAMBDA (Bitmap)

(* Gaska "20-Sep-88 08:45")

```

(* * Make a "cookie cutter" mask as used in ICONW)

```

```

(LET* [(Temp (BITMAPCREATE (IPLUS (BITMAPWIDTH Bitmap)
                                2)
                            (IPLUS (BITMAPHEIGHT Bitmap)
                                2)))
  (Mask (BITMAPCREATE (BITMAPWIDTH Bitmap)
                    (BITMAPHEIGHT Bitmap)
  (BITBLT Bitmap 0 0 Temp 1 1)
  (FILLREGION Temp (CREATEPOSITION 0 0)
    BLACKSHADE)
  (BITBLT Temp 1 1 Mask 0 0 (BITMAPWIDTH Bitmap)
    (BITMAPHEIGHT Bitmap)
    'INVERT
    'REPLACE)
  (BITBLT Bitmap 0 0 Mask 0 0 NIL NIL 'INPUT 'PAINT)
  Mask])

```

(EditBitmapMarkSpot

[LAMBDA (X/Position Y Window)

(* Gaska "20-Aug-88 10:11")

```

(* * Mark a point in a window)

```

```

(LET* ((X (if (POSITIONP X/Position)
              then (CAR X/Position)
              else X/Position))
  (Width (BITMAPWIDTH EditBitmapSpotMarker))
  (Height (BITMAPHEIGHT EditBitmapSpotMarker)))
(if (POSITIONP X/Position)
  then (SETQ Y (CDR X/Position)))
(BITBLT EditBitmapSpotMarker 0 0 Window (IDIFFERENCE X (IQUOTIENT Width 2))
  (IDIFFERENCE Y (IQUOTIENT Height 2))
  Width Height 'INPUT 'INVERT])

```

(EditBitmapMessage

[LAMBDA (Window Messages)

(* Gaska "26-Oct-88 09:48")

```

(* * Print a message)

```

```

(LET* [(OldWindow (WINDOWPROP Window 'MessageWindow))
  (MaxWidth (IMAX (IPLUS (STRINGWIDTH (for Message in (OR (LISTP Messages)
                                                            (LIST Messages))
                                                            largest (STRINGWIDTH Message EditBitmapMessageFont))
                    EditBitmapMessageFont)
                20)
  (fetch WIDTH of (WINDOWPROP Window 'REGION)
  (Height (IPLUS [ITIMES (FONTHEIGHT EditBitmapMessageFont)
                        (LENGTH (OR (LISTP Messages)
                                    (LIST Messages)
                        8))
  (MessageWindow (OR (if OldWindow
                        then (if (OR (ILESSP (fetch WIDTH of (WINDOWPROP OldWindow 'REGION))
                                        MaxWidth)
                                (ILESSP (fetch HEIGHT of (WINDOWPROP OldWindow 'REGION))
                                        Height))
                        then (DETACHWINDOW OldWindow)
                            (SHAPEW OldWindow (CREATEREGION (fetch LEFT
                                                                of (WINDOWPROP Window
                                                                    'REGION))
                                                                (fetch TOP of (WINDOWREGION Window))
                                                                [IMAX MaxWidth
                                                                (fetch WIDTH
                                                                of (WINDOWPROP Window
                                                                    'REGION)
                                                                Height))
                            (ATTACHWINDOW OldWindow Window 'TOP 'LEFT 'LOCALCLOSE))
                        OldWindow)
  (CREATEW (CREATEREGION (fetch LEFT of (WINDOWPROP Window 'REGION))
    (fetch TOP of (WINDOWREGION Window))
    MaxWidth Height]
  (if (NULL OldWindow)
    then (ATTACHWINDOW MessageWindow Window 'TOP 'LEFT 'LOCALCLOSE))

```

```

(WINDOWPROP Window 'MessageWindow MessageWindow)
(DSPFONT EditBitmapMessageFont MessageWindow)
(CLEARW MessageWindow)
(for Message in (OR (LISTP Messages)
                    (LIST Messages))
  do (PRIN1 Message MessageWindow)
      (TERPRI MessageWindow))

```

(EditBitmapMessageClose

```
[LAMBDA (Window) (* Gaska "26-Aug-88 09:58")
```

```
(* * Close the message window)
```

```
(CLOSEW (WINDOWPROP Window 'MessageWindow NIL))
```

(EditBitmapMoveRegion

```
[LAMBDA (Window Bitmap RegionGrid Grid MinWidth MinHeight) (* Gaska "31-Oct-88 07:32")
```

```
(* * Move a region)
```

```

(RESETLST
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (EditBitmapMessage Window "Select the region to be moved.")
  (LET* [(Region (EditBitmapGetWindowRegion Window RegionGrid MinWidth MinHeight))
        [NewBitmap (if Region
                        then (BITMAPCREATE (fetch WIDTH of Region)
                                           (fetch HEIGHT of Region))
                        (Position (if Region
                                     then (LET NIL (BITBLT (WINDOWPROP Window 'Bitmap)
                                                             (fetch LEFT of Region)
                                                             (fetch BOTTOM of Region)
                                                             NewBitmap 0 0 (fetch WIDTH of Region)
                                                             (fetch HEIGHT of Region))
                                     (EditBitmapMessage Window "Place the bitmap")
                                     (EditBitmapTrackBitmap Window NewBitmap Grid))
                                (CLOSEPROMPTWINDOW Window)
                                (if Position
                                  then (BLTSHADE WHITESHADE Bitmap (fetch LEFT of Region)
                                           (fetch BOTTOM of Region)
                                           (fetch WIDTH of Region)
                                           (fetch HEIGHT of Region)
                                           'REPLACE)
                                  (BITBLT NewBitmap 0 0 Bitmap (CAR Position)
                                           (CDR Position)
                                           (fetch WIDTH of Region)
                                           (fetch HEIGHT of Region)
                                           'INPUT
                                           (WINDOWPROP Window 'Operation))
                                  Bitmap)))]])

```

(EditBitmapPaintWindow

```
[LAMBDA (Window Bitmap BrushShape BrushSize Shade Operation Font) (* Gaska "4-Oct-88 09:33")
```

```
(* * Allows the user to paint with a brush)
```

```

(EditBitmapMessage Window '("Left button paints." "Middle button erases." "Right button gives a menu."
                           "Select QUIT to stop painting.))
(RESETLST
  (RESETSAVE NIL (LIST 'CURSOR (CURSOR)))
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (TOTOPW Window)
  (LET* ((Brush (\GETBRUSH (LIST BrushShape BrushSize)))
        (Stream (WINDOWPROP Window 'DSP))
        (Cursor (CURSORCREATE (BITMAPCREATE 16 16)
                               0 0))
        (Menu (create MENU
                      ITEMS _ ' (PAINT REPLACE INVERT ERASE QUIT)
                      MENUFONT _ Font
                      CENTERFLG _ T)))
        (BITBLT Brush 0 0 (EditBitmapCursorFields Cursor 'BITMAP)
                0 0 16 16)
        (LET* ((HotX (LRSH (IMIN (BITMAPWIDTH Brush)
                                16)
                          1))
              (HotY (LRSH (IMIN (BITMAPHEIGHT Brush)
                                16)
                          1)))
              (EditBitmapCursorFields Cursor 'HOTSPOT (CREATEPOSITION HotX HotY))
              (SETCURSOR Cursor)
              (bind Done until Done
                    do (GETMOUSESTATE)
                        (LET ((X (IDIFFERENCE (LASTMOUSEX Stream)
                                              HotX))

```

```

(Y (IDIFFERENCE (LASTMOUSEY Stream)
  HotY)))
(COND
  [(AND (LASTMOUSESTATE LEFT)
    (OR (EQ Operation 'REPLACE)
      (NEQ Shade BLACKSHADE)))]
    (COND
      ((EQ Operation 'REPLACE)
        (BITBLT Brush 0 0 Stream X Y NIL NIL 'INPUT 'ERASE)
        (BITBLT Brush 0 0 Bitmap X Y NIL NIL 'INPUT 'ERASE)
        (BITBLT Brush 0 0 Stream X Y NIL NIL 'MERGE 'PAINT Shade)
        (BITBLT Brush 0 0 Bitmap X Y NIL NIL 'MERGE 'PAINT Shade))
      ((EQ Operation 'ERASE)
        (BITBLT Brush 0 0 Stream X Y NIL NIL 'INPUT 'ERASE)
        (BITBLT Brush 0 0 Bitmap X Y NIL NIL 'INPUT 'ERASE))
      (T (BITBLT Brush 0 0 Stream X Y NIL NIL 'MERGE Operation Shade)
        (BITBLT Brush 0 0 Bitmap X Y NIL NIL 'MERGE Operation Shade)
        (LASTMOUSESTATE LEFT)
        (BITBLT Brush 0 0 Stream X Y NIL NIL 'INPUT Operation)
        (BITBLT Brush 0 0 Bitmap X Y NIL NIL 'INPUT Operation))
      ((LASTMOUSESTATE MIDDLE)
        (BITBLT Brush 0 0 Stream X Y NIL NIL 'INPUT 'ERASE)
        (BITBLT Brush 0 0 Bitmap X Y NIL NIL 'INPUT 'ERASE))
      ((LASTMOUSESTATE RIGHT)
        (LET ((Selection (MENU Menu)))
          (if (EQ Selection 'QUIT)
              then (SETQ Done T)
              else (SETQ Operation Selection)
            )
          (T NIL]
        T)))

```

(EditBitmapPaintWindowWithBitmap

[LAMBDA (Window Bitmap Pattern Operation Font)

(* Gaska "27-Oct-88 12:51")

(* * Allows the user to paint with a bitmap)

```

(EditBitmapMessage Window '("Left button paints." "Middle button erases." "Right button gives a menu."
  "Select QUIT to stop painting.))
(RESETLST
  (RESETSAVE NIL (LIST 'CURSOR (CURSOR)))
  (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
  (TOTOPW Window)
  [LET* ((Stream (WINDOWPROP Window 'DSP))
    (Cursor (CURSORCREATE (BITMAPCREATE 16 16)
      2 2))
    (Width (BITMAPWIDTH Pattern))
    (Height (BITMAPHEIGHT Pattern))
    (HalfWidth (LRSH Width 1))
    (HalfHeight (LRSH Height 1))
    (Buffer (BITMAPCREATE Width Height))
    (Buffer? NIL)
    (CursorX NIL)
    (CursorY NIL)
    (DrawX NIL)
    (DrawY NIL)
    (Menu (create MENU
      ITEMS _ ' (PAINT REPLACE INVERT ERASE QUIT)
      MENUFONT _ Font
      CENTERFLG _ T)))
    (BITBLT (\GETBRUSH ' (ROUND 4))
      0 0 (EditBitmapCursorFields Cursor 'BITMAP)
      0 0 16 16)
    (SETCURSOR Cursor)
    (bind Done until Done
      do (GETMOUSESTATE)
        [LET ((NewX (LASTMOUSEX Stream))
          (NewY (LASTMOUSEY Stream)))
          (if (AND (MOUSESTATE UP)
            (OR (NEQ NewX CursorX)
              (NEQ NewY CursorY)))
            then (if (AND Buffer? CursorX CursorY (EditBitmapInsideWindow Window CursorX CursorY)
              )
              then (BITBLT Buffer 0 0 Stream (IDIFFERENCE CursorX HalfWidth)
                (IDIFFERENCE CursorY HalfHeight)
                Width Height))
              (SETQ CursorX NewX)
              (SETQ CursorY NewY)
              (if (EditBitmapInsideWindow Window CursorX CursorY)
                then (BITBLT Stream (IDIFFERENCE CursorX HalfWidth)
                  (IDIFFERENCE CursorY HalfHeight)
                  Buffer 0 0 NIL NIL)
                  (BITBLT Pattern 0 0 Stream (IDIFFERENCE CursorX HalfWidth)
                    (IDIFFERENCE CursorY HalfHeight)
                    Width Height 'INPUT 'INVERT)
                    (SETQ Buffer? T)))
                (if (AND (MOUSESTATE (OR LEFT MIDDLE))

```

```

      (OR (NEQ NewX DrawX)
          (NEQ NewY DrawY)))
    then (SETQ DrawX NewX)
          (SETQ DrawY NewY)
          (SETQ Buffer? NIL)
          (COND
            ((AND (LASTMOUSESTATE LEFT)
                  (EQ Operation 'REPLACE))
              (BITBLT Pattern 0 0 Stream (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT 'ERASE)
              (BITBLT Pattern 0 0 Bitmap (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT 'ERASE)
              (BITBLT Pattern 0 0 Stream (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT 'PAINT)
              (BITBLT Pattern 0 0 Bitmap (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT 'PAINT))
            ((LASTMOUSESTATE LEFT)
              (BITBLT Pattern 0 0 Stream (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT Operation)
              (BITBLT Pattern 0 0 Bitmap (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT Operation))
            ((LASTMOUSESTATE MIDDLE)
              (BITBLT Pattern 0 0 Stream (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT 'ERASE)
              (BITBLT Pattern 0 0 Bitmap (IDIFFERENCE DrawX HalfWidth)
                      (IDIFFERENCE DrawY HalfHeight)
                      NIL NIL 'INPUT 'ERASE))
            (T NIL))
          else (if (LASTMOUSESTATE RIGHT)
                   then (LET ((Selection (MENU Menu)))
                        (if (EQ Selection 'QUIT)
                            then (SETQ Done T)
                            else (SETQ Operation Selection]
                        finally (if (EditBitmapInsideWindow Window CursorX CursorY)
                                then (BITBLT Buffer 0 0 Window (IDIFFERENCE CursorX HalfWidth)
                                     (IDIFFERENCE CursorY HalfHeight)
                                     Width Height]
                                T)))))

```

(EditBitmapPlaceBitmap

[LAMBDA (Window Bitmap Message Grid)

(* Gaska "23-Sep-88 12:25")

(* * Track and place a bitmap)

```

(if Message
  then (EditBitmapMessage Window Message))
(EditBitmapTrackBitmap Window Bitmap Grid])

```

(EditBitmapRandomBitWithinBrush

[LAMBDA (Diameter Brush)

(* Gaska " 2-Sep-88 08:12")

(* * Get a random bit that falls within a circular brush)

```

(bind X Y until X do (LET ((RandX (RAND 1 Diameter))
                          (RandY (RAND 1 Diameter)))
  (if (EQ (BITMAPBIT Brush RandX RandY)
          1)
    then (SETQ X RandX)
         (SETQ Y RandY)))
finally (RETURN (CONS X Y])

```

(EditBitmapReadBitmap

[LAMBDA (Stream)

(* edited%: "31-Aug-88 13:04")

(* * Reads a bitmap)

```

(LET* ((Width (BIN16 Stream))
      (Height (BIN16 Stream))
      (Bitmap (BITMAPCREATE Width Height)))
  (\BINS Stream (fetch BITMAPBASE of Bitmap)
    0
    (ITIMES (fetch BITMAPRASTERWIDTH of Bitmap)
      Height 2))
  Bitmap])

```

(EditBitmapRemoveConnectedPixels


```

(NewBitmap (if Region
            then (BITMAPCREATE (fetch WIDTH of Region)
                               (fetch HEIGHT of Region]
(EditBitmapMessageClose Window)
(if Region
    then (BITBLT Bitmap (fetch LEFT of Region)
                  (fetch BOTTOM of Region)
                  NewBitmap 0 0 (fetch WIDTH of Region)
                  (fetch HEIGHT of Region))
    NewBitmap])

```

(EditBitmapTrackBitmap

[LAMBDA (Window Bitmap Grid)

(* Gaska "31-Oct-88 07:34")

(* * Tracks a bitmap until a button goes down and comes up)

```

(LET* ((Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (Buffer (BITMAPCREATE Width Height))
      (DisplayStream (GETSTREAM Window))
      (MaximumX (IDIFFERENCE (WINDOWPROP Window 'WIDTH)
                              Width))
      (MaximumY (IDIFFERENCE (WINDOWPROP Window 'HEIGHT)
                              Height))
      (X NIL)
      (Y NIL))
(TOTOPW Window)
(until (LET NIL (GETMOUSESTATE)
          (EditBitmapInsideWindow Window (LASTMOUSEX DisplayStream)
                                   (LASTMOUSEY DisplayStream)))
      do NIL)
(until (LASTMOUSESTATE (NOT UP))
      do (GETMOUSESTATE)
        [LET ((NewX (EditBitmapGetGriddedValue (LASTMOUSEX DisplayStream)
                                                Grid))
              (NewY (EditBitmapGetGriddedValue (LASTMOUSEY DisplayStream)
                                                Grid)))
          (if (OR (NEQ NewX X)
                  (NEQ NewY Y))
              then (if (AND X Y (EditBitmapInsideWindow Window X Y))
                      then (BITBLT Buffer 0 0 Window X Y Width Height))
                      (SETQ X NewX)
                      (SETQ Y NewY)
                      (if (EditBitmapInsideWindow Window X Y)
                          then (BITBLT Window NewX NewY Buffer 0 0 NIL NIL)
                          (BITBLT Bitmap 0 0 DisplayStream NewX NewY Width Height 'INPUT)
                          'PAINT]
              finally (if (EditBitmapInsideWindow Window X Y)
                          then (BITBLT Buffer 0 0 Window X Y Width Height)
                          (RETURN (CREATEPOSITION X Y]))

```

(EditBitmapTrim

[LAMBDA (Bitmap)

(* Gaska "26-Sep-88 08:36")

(* * Remove the white space from the edges of a bitmap)

```

(LET* [(Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      [Left (for X from 0 to (SUB1 Width) thereis (for Y from 0 to (SUB1 Height)
                                                    thereis (EQ 1 (BITMAPBIT Bitmap X Y))
      [Right (if Left
                then (for X from (SUB1 Width) to 0 by -1
                      thereis (for Y from 0 to (SUB1 Height) thereis (EQ 1 (BITMAPBIT Bitmap X Y))
      [Top (if Left
              then (for Y from (SUB1 Height) to 0 by -1
                    thereis (for X from 0 to (SUB1 Width) thereis (EQ 1 (BITMAPBIT Bitmap X Y))
      [Bottom (if Top
                then (for Y from 0 to (SUB1 Height) thereis (for X from 0 to (SUB1 Width)
                                                              thereis (EQ 1 (BITMAPBIT Bitmap X Y))
      (if (AND Left Right Top Bottom (NEQ (ADD1 (IDIFFERENCE Right Left))
                                           Width)
          (NEQ (ADD1 (IDIFFERENCE Top Bottom))
               Height))
          then (LET [(NewBitmap (BITMAPCREATE (ADD1 (IDIFFERENCE Right Left))
                                                (ADD1 (IDIFFERENCE Top Bottom))
              (BITBLT Bitmap Left Bottom NewBitmap 0 0 (ADD1 (IDIFFERENCE Right Left))
              (ADD1 (IDIFFERENCE Top Bottom)))
              NewBitmap])

```

(EditBitmapVerticesFromRegion

[LAMBDA (Region)

(* Gaska "11-Aug-88 14:43")

(* * Make a list of vertices from a region)

```
(LIST (CONS (fetch LEFT of Region)
            (fetch BOTTOM of Region))
      (CONS (fetch RIGHT of Region)
            (fetch BOTTOM of Region))
      (CONS (fetch RIGHT of Region)
            (fetch TOP of Region))
      (CONS (fetch LEFT of Region)
            (fetch TOP of Region))
      (CONS (fetch LEFT of Region)
            (fetch BOTTOM of Region)))
```

(EditBitmapWireInput

[LAMBDA (Window Grid Orthogonal? Closed?)

(* Gaska "15-Sep-88 14:15")

(* * Fetch a series of points for a wire)

```
(RESETLST
 (RESETSAVE NIL (LIST 'EditBitmapMessageClose Window))
 (EditBitmapMessage Window '("Enter the points using left button." "Middle button erases last fixed
                             point." "Left click outside the window to stop."))
 (LET [(PointList (EditBitmapGetPointList Window Grid Orthogonal? (if Closed?
                                                                    then 'EditBitmapWireShowClosed
                                                                    else 'EditBitmapWireShowOpen))
      (if (IGEQ (LENGTH PointList)
                2)
          then PointList))]))
```

(EditBitmapWireShowClosed

[LAMBDA (Window X Y Previous)

(* Gaska "22-Aug-88 08:18")

(* * Provides the rubberbanding feedback for the user inputting a point for an open wire.)

```
(EditBitmapShowPoint Window X Y)
(AND Previous (DRAWLINE (CAAR Previous)
                        (CDAR Previous)
                        X Y 1 'INVERT Window))
(AND (CDR Previous)
     (DRAWLINE (CAAR (LAST Previous))
               (CDAR (LAST Previous))
               X Y 1 'INVERT Window))
```

(EditBitmapWireShowOpen

[LAMBDA (Window X Y Previous)

(* Gaska "22-Aug-88 08:18")

(* * Provides the rubberbanding feedback for the user inputting a point for an open wire.)

```
(EditBitmapShowPoint Window X Y)
(AND Previous (DRAWLINE (CAAR (LAST Previous))
                        (CDAR (LAST Previous))
                        X Y 1 'INVERT Window))
```

(EditBitmapWriteBitmap

[LAMBDA (Stream Bitmap)


(* edited%: "31-Aug-88 13:13")

(* * Write a bitmap)


```
(LET* ((Width (BITMAPWIDTH Bitmap))
      (Height (BITMAPHEIGHT Bitmap))
      (BOUT16 Stream Width)
      (BOUT16 Stream Height)
      (\BOUTS Stream (fetch BITMAPBASE of Bitmap)
                     0
                     (ITIMES (fetch BITMAPRASTERWIDTH of Bitmap)
                              Height 2))
      Bitmap])
```

)

(* * Cursors)

```
(RPAQ EditBitmapCircleCenter (CURSORCREATE ' 
                                'NIL 7 7))
```

```
(RPAQ EditBitmapCircleEdge (CURSORCREATE ' 
                                'NIL 15 7))
```

```
(RPAQ EditBitmapEllipseCenter (CURSORCREATE ' 
                                'NIL 7 7))
```

```
(RPAQ EditBitmapEllipseMajor (CURSORCREATE ' 
                                     'NIL 15 7))
```

```
(RPAQ EditBitmapEllipseMinor (CURSORCREATE ' 
                                     'NIL 7 15))
```

```
(RPAQ EditBitmapMagnifyCursor (CURSORCREATE ' 
                                     'NIL 7 7))
```

(* * Variables)

```
(RPAQQ EditBitmapAirbrushTimerIntervals ((FAST 0)
                                           (MEDIUM 75)
                                           (SLOW 200)))
```

```
(RPAQQ EditBitmapBlockTime 5)
```

```
(RPAQQ EditBitmapDefaultAirBrushSize 8)
```

```
(RPAQQ EditBitmapDefaultArrowHeight 15)
```

```
(RPAQQ EditBitmapDefaultArrowWidth 10)
```

```
(RPAQQ EditBitmapDefaultAutoSaveDeltaTime 10)
```

```
(RPAQQ EditBitmapDefaultAveraging ((0 2 0 2 9 2 0 2 0)
                                     13))
```

```
(RPAQQ EditBitmapDefaultBrushSize 1)
```

```
(RPAQQ EditBitmapDefaultFont (HELVETICA 12 MRR))
```

```
(RPAQQ EditBitmapDefaultListOfDashings ((6 3 1 3)
                                           (4 2)
                                           (3 1)))
```

```
(RPAQQ EditBitmapDefaultListOfTextures (65535 42405 34850 18450 8580 5160 5140 32800 0))
```

```
(RPAQQ EditBitmapDefaultPaintBrushSize 2)
```

```
(RPAQQ EditBitmapMaxArrowSize 100)
```

```
(RPAQQ EditBitmapMaxBrushSize 128)
```

```
(RPAQQ EditBitmapMaxGridSize 200)
```

```
(RPAQQ EditBitmapMaxPatternSize 250)
```

```
(RPAQQ EditBitmapMenuFont (HELVETICA 12 BRR))
```

```
(RPAQQ EditBitmapMenuItems
[("Paint" (EditBitmapChooseSubitem)
  "Invoke a paint editor"
  (SUBITEMS ("Bitmap Editor Paint with Brush" EditBitmapPaintWithBrush "Use the Bitmap Editor paint
    facility with a brush")
    ("Bitmap Editor Paint with Airbrush" EditBitmapPaintWithAirbrush "Use the Bitmap Editor
    paint facility with a airbrush")
    ("Bitmap Editor Paint with Pattern" EditBitmapPaintWithBitmap "Use the Bitmap Editor paint
    facility with a bitmap"))))
("Edit" (EditBitmapChooseSubitem)
  "Invoke an editor"
  (SUBITEMS ("Magnifier Pixel Editor" EditBitmapMagnifyEdit "Invoke a 16 X 16 bit magnifier pixel
    editor")
    ("Pixel Edit Bitmap" EditBitmapPixelEditWindow "Invoke the bitmap editor to precisely edit
    the bitmap.")
    ("Pixel Edit Region" EditBitmapPixelEditRegion "Invoke the bitmap editor to precisely edit
    a region within the bitmap")
    ("Edit Region" EditBitmapBitmapEditRegion "Start another bitmap editor on a region"))))
("Copy Operations" (EditBitmapChooseSubitem)
  "Bitmap copy operations"
  (SUBITEMS ("Create New Copy of Bitmap" EditBitmapCopyCreate "Create a copy of the main bitmap")
    ("Edit Copy of Bitmap" EditBitmapCopyEdit "Edit the previously saved copy of the bitmap")
    ("Apply Operation to Copy and Original" EditBitmapCopyOperate "Apply the currently
    selected operation upon the copy and the original"))))
("Draw Lines" EditBitmapDrawLines "Draw one or more line segments")
("Draw Arrow" EditBitmapDrawArrowheadOpen "Draw an open arrowhead at the end of one or more line
  segments" (SUBITEMS ("Draw Filled Arrow" EditBitmapDrawArrowheadFilled "Draw a filled arrowhead
    at the end of one or more line segments"))))
("Draw Rectangle" EditBitmapDrawBox "Draw a rectangle")
("Draw Polygon" EditBitmapDrawPolygon "Draw a polygon")
("Draw Circle" EditBitmapDrawCircle "Draw a circle")])
```

```

("Draw Ellipse" EditBitmapDrawEllipse "Draw an ellipse")
("Draw Open Curve" EditBitmapDrawCurveOpen "Draw an open curve")
("Draw Closed Curve" EditBitmapDrawCurveClosed "Draw a closed curve")
("Draw Text" EditBitmapDrawText "Draw text")
("Draw Grid" EditBitmapDrawGridWindow "Cover the bitmap with a grid" (SUBITEMS ("Draw Grid in Region"
    EditBitmapDrawGridRegion
    "Draw a grid in a region
    of the bitmap"))))

["Place Pattern" EditBitmapPlacePattern "Place copies of a pattern"
    (SUBITEMS ("Pattern Array" EditBitmapPatternArray "Place an array of patterns")
        ("Tessellate" EditBitmapTessellateBitmap "Cover the bitmap with a repeated pattern"
            (SUBITEMS ("Tessellate Region" EditBitmapTessellateRegion "Cover a region with a
                repeated pattern"]

("Follow Pixels" (EditBitmapChooseSubitem)
    "Erase or draw over all connected 'on' pixels from a given starting region or point"
    (SUBITEMS ("Erase Connected Pixels from Region" EditBitmapErasePixelsRegion "Erase all connected
        'on' pixels from a given starting region")
        ("Erase Connected Pixels from Selected Pixel" EditBitmapErasePixelsSingle "Erase all
            connected 'on' pixels from a given starting pixel")
        ("Draw Over Connected Pixels from Region" EditBitmapDrawOverPixelsRegion "Draw over all
            connected 'on' pixels from a given starting region")
        ("Draw Over Connected Pixels from Selected Pixel" EditBitmapDrawOverPixelsSingle "Draw
            over all connected 'on' pixels from a given starting pixel"))))

("Input From Screen" EditBitmapAddBitmap "Add a bitmap from the screen")
("Move a Region" EditBitmapMoveRectangularRegion "Move a rectangular region in the bitmap")
("Transform" (EditBitmapChooseSubitem)
    "Subitems transform the bitmap in various ways."
    (SUBITEMS ("Distort Region" EditBitmapDistort "Distort a rectangular region")
        ("Invert" EditBitmapInvertBitmap "Invert the bitmap")
        ("Shift" EditBitmapShiftBitmap "Shift the bitmap")
        ("Rotate" EditBitmapRotateBitmap "Rotate the bitmap")
        ("Shrink" EditBitmapShrinkBitmap "Shrink the bitmap")
        ("Expand" EditBitmapExpandBitmap "Expand the Bitmap")
        ("Smaller" EditBitmapMakeSmaller "Make the bitmap smaller by sweeping out a region")
        ("Exact Size" EditBitmapMakeExact "Make the bitmap smaller to an exact size"))))

["Coloring" (EditBitmapChooseSubitem)
    "Subitems manipulate the coloring of the bitmap in various ways"
    (SUBITEMS ("Add Texture" EditBitmapAddTexture "Add texture to the bitmap")
        ("Fill Region(s)" EditBitmapFillRegionDefault "Fill white region(s) of the bitmap with the
            default color." (SUBITEMS ("Fill Region(s) with Specified Color"
                EditBitmapFillRegionSpecified "Fill white
                region(s) of the bitmap with a specified color."))
        )
        ("Fill Box" EditBitmapFillBox "Fills a box with the default color" (SUBITEMS ("Fill Box
            with
            Specified
            Color"
                EditBitmapFillBoxSpecified
                "Fills
                a box
                with a
                specif
                color"
                )))
        ("Switch Colors" EditBitmapInvertColor "Switch black for white and white for black in the
            bitmap." (SUBITEMS ("Switch Colors in Region" EditBitmapInvertColorRegion
                "Switch black for white and white for black in a region")
            ))
        ("Averaging" EditBitmapAveraging "Apply an averaging array to the bitmap"
            (SUBITEMS ("Show Specification Format" (EditBitmapShowAveragingFormat)
                "Show the format of the averaging specification.")))
        ("Mask" EditBitmapMask "Create a mask as used in ICONW")
        ("Clear" EditBitmapClearBitmap "Set the bitmap to be all white." (SUBITEMS ("Clear
            Region"
                EditBitmapClearRegion
                "Clear a
                region
                to all
                white.")
            ("Clear All But
                Region"
                EditBitmapClearAllButRegion
                "Clear the
                entire
                bitmap
                except for
                the given
                region to
                white"]

("Edges" (EditBitmapChooseSubitem)
    "Remove white space or add a border to the bitmap"
    (SUBITEMS ("Trim White Space" EditBitmapTrimBitmap "Remove the white space from the edges of the
        bitmap")
        ("Trim Sides Precisely" EditBitmapTrimSides "Trim bits from one or more sides of the

```

ied

```

        bitmap")
        ("Add Border" EditBitmapAddBorder "Add a border to the bitmap"))
    ("Pattern Management" (EditBitmapChooseSubitem)
        "Subitems add/delete/edit patterns"
        (SUBITEMS ("Add pattern" (EditBitmapAddPattern)
            "Add a pattern")
            ("Delete Pattern" (EditBitmapDeletePattern)
                "Delete a pattern")
            ("Store Patterns" (EditBitmapStorePatterns)
                "Store the patterns on a file")
            ("Retrieve Patterns" (EditBitmapFetchPatterns)
                "Read patterns from a file")
            ("Trim Pattern" (EditBitmapTrimPattern)
                "Remove the white space from the edges of the pattern")
            ("Copy/Edit Pattern" (EditBitmapCopyEditPattern)
                "Make a copy of a pattern and edit the copy")
            ("Edit Pattern" (EditBitmapEditPattern)
                "Edit a pattern"))))
    ("Undo" (EditBitmapUndo)
        "Undo the last editing operation")
    ("Save/Restore" (EditBitmapChooseSubitem)
        "Save/restore bitmap"
        (SUBITEMS ("Auto-Save Status" (EditBitmapShowAutoSave)
            "Display auto-save parameters"
            (SUBITEMS ("Set Auto-Save" (EditBitmapSetAutoSave)
                "Turn auto-save on or off")
                ("Auto-Save File Name" (EditBitmapSetAutoSaveFile)
                    "Specify the auto-save file name")
                ("Auto-Save Interval" (EditBitmapSetAutoSaveInterval)
                    "Specify the auto-save time interval"))))
            ("Checkpoint Save" (EditBitmapCheckpointSave)
                "Save the current state of the bitmap")
            ("Checkpoint Restore" EditBitmapCheckpointRestore "Restore bitmap to last checkpoint")
            ("Reset" EditBitmapResetBitmap "Reset the bitmap to the beginning of the edit session.")
            ("Put to File" (EditBitmapFileSave)
                "Write the bitmap to a file")
            ("Get From File" EditBitmapFileRestore "Read a bitmap from a file"))))
    ("Insert Bitmap" (EditBitmapChooseSubitem)
        "Insert the bitmap into TEdit or SKETCH"
        (SUBITEMS ("Insert Bitmap Into TEdit" EditBitmapInsertBitmapIntoTedit "Insert the bitmap into a
            TEdit window at the current insertion point."
            ("Insert Bitmap Into SKETCH" EditBitmapInsertBitmapIntoSketch "Insert the bitmap into a
                SKETCH window.")))
    ("Parameters" (EditBitmapShowParameters)
        "Display the editing parameters"
        (SUBITEMS ("Drawing Brush Size" (EditBitmapSetDrawBrushSize)
            "Set the drawing brush size")
            ("Paint Brush Size" (EditBitmapSetPaintBrushSize)
                "Set the paint brush size")
            ("Air Brush Size" (EditBitmapSetAirBrushSize)
                "Set the airbrush size")
            ("Air Brush Speed" (EditBitmapSetAirBrushSpeed)
                "Set the airbrush speed")
            ("Brush Shape" (EditBitmapSetBrushShape)
                "Set the brush shape")
            ("Dashing" (EditBitmapSetDashing)
                "Set the dashing")
            ("Operation" (EditBitmapSetOperation)
                "Set the drawing operation")
            ("Texture" (EditBitmapSetShade)
                "Set the texture" "Set the shade (texture)")
            ("Font" (EditBitmapSetFont)
                "Set the font")
            ("Arrowhead" (EditBitmapSetArrowhead)
                "Set the arrowhead width and height")
            ("Region Selection Grid" (EditBitmapSetRegionGrid)
                "Set gridding for region selection")
            ("Grid" (EditBitmapSetGrid)
                "Set gridding")
            ("Orthogonal Lines" (EditBitmapSetOrthogonal)
                "Turn orthogonal line drawing on or off"]])

```

(RPAQQ EditBitmapMenuPointer



(RPAQQ EditBitmapMessageFont (GACHA 12 BRR))

(RPAQQ EditBitmapMinArrowSize 5)

(RPAQQ EditBitmapMinPatternSize 4)

(RPAQQ EditBitmapMinRegionSize 10)

(RPAQQ EditBitmapMinSize 10)

(RPAQQ EditBitmapPixelSelectBoxSize 9)

(RPAQQ **EditBitmapSpotMarker** ,

(DECLARE%: DOEVAL@COMPILE DONTCOPY

```
(GLOBALVARS EditBitmapAirbrushTimerIntervals EditBitmapBlockTime EditBitmapCircleCenter EditBitmapCircleEdge
  EditBitmapDefaultAirBrushSize EditBitmapDefaultArrowHeight EditBitmapDefaultArrowWidth
  EditBitmapDefaultAutoSaveDeltaTime EditBitmapDefaultAveraging EditBitmapDefaultBrushSize
  EditBitmapDefaultFont EditBitmapDefaultListOfDashings EditBitmapDefaultListOfTextures
  EditBitmapDefaultPaintBrushSize EditBitmapEllipseCenter EditBitmapEllipseMajor EditBitmapEllipseMinor
  EditBitmapListOfDashings EditBitmapListOfTextures EditBitmapMagnifyCursor EditBitmapMaxArrowSize
  EditBitmapMaxBrushSize EditBitmapMaxGridSize EditBitmapMaxPatternSize EditBitmapMenuFont
  EditBitmapMenuItems EditBitmapMenuPointer EditBitmapMessageFont EditBitmapMinArrowSize
  EditBitmapMinPatternSize EditBitmapMinRegionSize EditBitmapMinSize EditBitmapPixelSelectBoxSize
  EditBitmapSpotMarker)
)
```

(* * Auxiliary files)

```
(FILESLOAD READNUM (SYSLOAD FROM LISPUSERS)
  FILLREGION)
```

(* * Initialization)

(SETQ EditBitmapListOfTextures (COPYALL EditBitmapDefaultListOfTextures))

(SETQ EditBitmapListOfDashings (COPYALL EditBitmapDefaultListOfDashings))

FUNCTION INDEX

EditBitmap	2	EditBitmapEllipseShowMajor	48
EditBitmapAddBitmap	6	EditBitmapEllipseShowMinor	48
EditBitmapAddBorder	6	EditBitmapErasePixelsRegion	27
EditBitmapAddPattern	30	EditBitmapErasePixelsSingle	27
EditBitmapAddTexture	6	EditBitmapExpandBitmap	12
EditBitmapAirbrushOutline	41	EditBitmapExpandQuadrilateral	34
EditBitmapAirbrushPaint	41	EditBitmapFetchPatterns	31
EditBitmapArrowhead	43	EditBitmapFileRestore	13
EditBitmapAutoSave	23	EditBitmapFileSave	13
EditBitmapAverageBitInBitmap	22	EditBitmapFillBox	13
EditBitmapAveraging	22	EditBitmapFillBoxSpecified	13
EditBitmapBitmapAverage	22	EditBitmapFillRegion	13
EditBitmapBitmapDistort	33	EditBitmapFillRegionDefault	14
EditBitmapBitmapEditRegion	6	EditBitmapFillRegionSpecified	14
EditBitmapBitmapFromFile	6	EditBitmapFollowPixelsMatrix	27
EditBitmapBitmapInvertDiagonal	43	EditBitmapFollowPixelsMatrixClose	28
EditBitmapBitmapInvertHorizontal	43	EditBitmapFollowPixelsMatrixDone	28
EditBitmapBitmapInvertVertical	44	EditBitmapFollowPixelsMatrixSelect	28
EditBitmapBitmapRotate	44	EditBitmapFollowPixelsRegion	28
EditBitmapBitmapRotateArbitrary	44	EditBitmapFollowPixelsSingle	29
EditBitmapBitmapShift	45	EditBitmapFontStylesheet	31
EditBitmapBlock?	7	EditBitmapFromScreen	49
EditBitmapBorder	45	EditBitmapGetAveragingInfo	22
EditBitmapBoxInput	45	EditBitmapGetDeltas	35
EditBitmapBySections	2	EditBitmapGetExtents	35
EditBitmapChangeBitmapSize	7	EditBitmapGetFileName	49
EditBitmapChangeBitmapSizeMaybe	7	EditBitmapGetGriddedValue	49
EditBitmapCheckpointRestore	7	EditBitmapGetNewDashing	36
EditBitmapCheckpointSave	7	EditBitmapGetNewVertices	35
EditBitmapChooseSubitem	3	EditBitmapGetPattern	31
EditBitmapCircleGetRadiusPoint	45	EditBitmapGetPoint	49
EditBitmapCircleInput	45	EditBitmapGetPointAnywhere	50
EditBitmapCircleShow	45	EditBitmapGetPointList	50
EditBitmapClearAllButRegion	7	EditBitmapGetPosition	51
EditBitmapClearBitmap	8	EditBitmapGetQuadrilateral	35
EditBitmapClearRegion	8	EditBitmapGetScreenBitmap	51
EditBitmapClose	3	EditBitmapGetScreenRegion	51
EditBitmapConfirmDashing	36	EditBitmapGetShade	14
EditBitmapConfirmDiscard	3	EditBitmapGetSizeFactor	14
EditBitmapConstrainRegion	46	EditBitmapGetStringFromUser	51
EditBitmapCopyBitmap	46	EditBitmapGetWindowRegion	51
EditBitmapCopyCreate	8	EditBitmapImageObjButtonEventInFn	40
EditBitmapCopyEdit	8	EditBitmapImageObjectCreate	40
EditBitmapCopyEditPattern	30	EditBitmapInsertBitmapIntoSketch	40
EditBitmapCopyOperate	8	EditBitmapInsertBitmapIntoTedit	41
EditBitmapCreatePixelSelectionCursor	46	EditBitmapInsideWindow	52
EditBitmapCursorFields	46	EditBitmapInsureGriddedRegion	52
EditBitmapCurveInput	47	EditBitmapInvertBitmap	14
EditBitmapDeletePattern	30	EditBitmapInvertColor	15
EditBitmapDisplayBitmap	3	EditBitmapInvertColorRegion	15
EditBitmapDistance	47	EditBitmapLocatePixelOn	52
EditBitmapDistort	33	EditBitmapMagnify	24
EditBitmapDistortBitmap	34	EditBitmapMagnifyBitmap	25
EditBitmapDoEditItem	3	EditBitmapMagnifyChange	25
EditBitmapDoInversion	9	EditBitmapMagnifyClose	25
EditBitmapDoRotation	9	EditBitmapMagnifyEdit	25
EditBitmapDrawArrowhead	9	EditBitmapMagnifyEditButtonEventFn	26
EditBitmapDrawArrowheadFilled	9	EditBitmapMagnifyMoved	26
EditBitmapDrawArrowheadOpen	10	EditBitmapMagnifySelectButtonEventFn	26
EditBitmapDrawBox	10	EditBitmapMakeDirty	14
EditBitmapDrawCircle	10	EditBitmapMakeExact	5
EditBitmapDrawCurve	10	EditBitmapMakeGrid	52
EditBitmapDrawCurveClosed	10	EditBitmapMakeMask	53
EditBitmapDrawCurveOpen	11	EditBitmapMakeMenu	4
EditBitmapDrawEllipse	11	EditBitmapMakePattern	31
EditBitmapDrawGrid	11	EditBitmapMakeSmaller	15
EditBitmapDrawGridRegion	11	EditBitmapMakeUndirty	4
EditBitmapDrawGridWindow	11	EditBitmapMarkSpot	53
EditBitmapDrawLines	11	EditBitmapMask	15
EditBitmapDrawOverConnectedPixels	47	EditBitmapMenuSelectionFn	4
EditBitmapDrawOverPixels	26	EditBitmapMessage	53
EditBitmapDrawOverPixelsRegion	27	EditBitmapMessageClose	54
EditBitmapDrawOverPixelsSingle	27	EditBitmapMoveRectangularRegion	15
EditBitmapDrawPolygon	12	EditBitmapMoveRegion	54
EditBitmapDrawQuadrilateral	34	EditBitmapOperationMenu	36
EditBitmapDrawText	12	EditBitmapPaintWindow	54
EditBitmapEditPattern	30	EditBitmapPaintWindowWithBitmap	55
EditBitmapEditRegion	47	EditBitmapPaintWithAirbrush	16
EditBitmapEllipseGetMajorPoint	48	EditBitmapPaintWithBitmap	16
EditBitmapEllipseGetMinorPoint	48	EditBitmapPaintWithBrush	16
EditBitmapEllipseInput	48	EditBitmapPatternArray	16
EditBitmapEllipseOrientation	48	EditBitmapPatternMenu	32


```
{MEDLEY}<notecards>library>bitmapeditor.;1
```

EditBitmapPixelEditRegion	17	EditBitmapSetPatternMenu	19
EditBitmapPixelEditWindow	17	EditBitmapSetQuit	5
EditBitmapPlaceBitmap	56	EditBitmapSetRegionGrid	39
EditBitmapPlacePattern	17	EditBitmapSetShade	39
EditBitmapPopUpMenu	4	EditBitmapSetSide	19
EditBitmapQuit	4	EditBitmapSetSideDone	19
EditBitmapRandomBitWithinBrush	56	EditBitmapSetSideMenu	19
EditBitmapReadBitmap	56	EditBitmapSetStop	5
EditBitmapRemoveConnectedPixels	56	EditBitmapSetWindowProps	5
EditBitmapRemovePixels	29	EditBitmapShiftBitmap	19
EditBitmapRepaintFn	5	EditBitmapShowAutoSave	24
EditBitmapResetBitmap	17	EditBitmapShowAveragingFormat	23
EditBitmapReshapeWindow	57	EditBitmapShowParameters	40
EditBitmapRotateBitmap	18	EditBitmapShowPattern	32
EditBitmapRound	57	EditBitmapShowPoint	57
EditBitmapSaveBitmap	18	EditBitmapShrinkBitmap	20
EditBitmapScreenInput	57	EditBitmapSmaller	57
EditBitmapSelectBrushOffset	29	EditBitmapStop	6
EditBitmapSelectTargetWindow	41	EditBitmapStorePatterns	32
EditBitmapSetAirBrushSize	37	EditBitmapTessellate	20
EditBitmapSetAirBrushSpeed	37	EditBitmapTessellateBitmap	20
EditBitmapSetArrowhead	37	EditBitmapTessellateRegion	21
EditBitmapSetAutoSave	23	EditBitmapTrackBitmap	58
EditBitmapSetAutoSaveFile	24	EditBitmapTrim	58
EditBitmapSetAutoSaveInterval	24	EditBitmapTrimBitmap	21
EditBitmapSetBrushShape	37	EditBitmapTrimPattern	33
EditBitmapSetDashing	37	EditBitmapTrimSides	21
EditBitmapSetDrawBrushSize	38	EditBitmapUndo	22
EditBitmapSetError	5	EditBitmapVerticesFromRegion	58
EditBitmapSetFont	38	EditBitmapWaitForFinished	6
EditBitmapSetGrid	38	EditBitmapWireInput	59
EditBitmapSetOperation	38	EditBitmapWireShowClosed	59
EditBitmapSetOrthogonal	39	EditBitmapWireShowOpen	59
EditBitmapSetPaintBrushSize	39	EditBitmapWriteBitmap	59
EditBitmapSetPatternAttribute	18		
EditBitmapSetPatternDone	18		

VARIABLE INDEX

BITMAPEDITORCOMS	1	EditBitmapEllipseMinor	60
EditBitmapAirbrushTimerIntervals	60	EditBitmapMagnifyCursor	60
EditBitmapBlockTime	60	EditBitmapMaxArrowSize	60
EditBitmapCircleCenter	59	EditBitmapMaxBrushSize	60
EditBitmapCircleEdge	59	EditBitmapMaxGridSize	60
EditBitmapDefaultAirBrushSize	60	EditBitmapMaxPatternSize	60
EditBitmapDefaultArrowHeight	60	EditBitmapMenuFont	60
EditBitmapDefaultArrowWidth	60	EditBitmapMenuItems	60
EditBitmapDefaultAutoSaveDeltaTime	60	EditBitmapMenuPointer	62
EditBitmapDefaultAveraging	60	EditBitmapMessageFont	62
EditBitmapDefaultBrushSize	60	EditBitmapMinArrowSize	62
EditBitmapDefaultFont	60	EditBitmapMinPatternSize	62
EditBitmapDefaultListOfDashings	60	EditBitmapMinRegionSize	62
EditBitmapDefaultListOfTextures	60	EditBitmapMinSize	62
EditBitmapDefaultPaintBrushSize	60	EditBitmapPixelSelectBoxSize	62
EditBitmapEllipseCenter	59	EditBitmapSpotMarker	63
EditBitmapEllipseMajor	60		
