```
--File:  WManWindows.mesa
--Edited by Sandman       October 7, 1977  9:46 AM

DIRECTORY
  SegmentDefs:  FROM "segmentdefs",
  StreamDefs:  FROM "streamdefs",
  StringDefs:  FROM "stringdefs",
  SystemDefs:  FROM "systemdefs",
  MenuDefs:  FROM "menudefs",
  RectangleDefs:  FROM "rectangledefs",
  WindowDefs:  FROM "windowdefs",
  WManagerDefs:  FROM "wmanagerdefs";

DEFINITIONS FROM StreamDefs, MenuDefs, RectangleDefs, WindowDefs, WManagerDefs;

WManWindows:  PROGRAM[WMState:  WMDataHandle]
  IMPORTS SegmentDefs, StreamDefs, SystemDefs, MenuDefs, RectangleDefs, WindowDefs, WManagerDefs
  EXPORTS WManagerDefs
  SHARES StreamDefs, WManagerDefs =

BEGIN

OPEN WMState;

CR: CHARACTER = 15C;

-- some externals

PutSelect: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord]=
  BEGIN
  -- Declare Locals
  i: CARDINAL;
  str: STRING;
  nw:  WindowHandle;
  doit:  BOOLEAN;
  [nw, doit] ← WindowFrontEnd[w, leftbutton];
  IF doit THEN
    BEGIN
    -- get current selection and jam it into the keyboard stream
    str ← GetSelection[w];
    IF nw.ks # NIL THEN
      FOR i DECREASING IN [0..str.length) DO
        nw.ks.putback[nw.ks, str[i]];
        ENDLOOP;
    SystemDefs.FreeHeapString[str];
    END;
  ButtonWait;
  SetCursor[textpointer];
  END;

  CreateWindow: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord] =
    BEGIN
    -- define locals
      i: INTEGER;
      rectangle: Rptr;
      ds: DisplayHandle;
      ks: StreamHandle;
      name: STRING;
      nw: WindowHandle;
      mb: AMouseButton ← None;
    -- Look at mouse for what to do
      SetCursor[leftbutton];
      UNTIL (mb ← GetMouseButton[]) = Red DO
        IF mb = Blue THEN
          BEGIN
          ButtonWait;
          SetCursor[textpointer];
          RETURN;
          END;
        ENDLOOP;
    -- now create  a new one
      SetCursor[hourglass];
      ks ← StreamDefs.CreateKeyStream[];
      [x,y] ← CursorToMapCoords[defaultmapdata, xcursorloc↑, ycursorloc↑];
      rectangle ← CreateRectangle[defaultmapdata, x, 200, y, 200];
      [name, i] ← AssignScratchFile[];
```

```
    ds ← CreateDisplayStream[rectangle];
    nw ← CreateDisplayWindow[scratch, rectangle, ds, ks, name];
    scratchfiles[i] ← nw.file;
    SystemDefs.FreeHeapString[name];
    StreamDefs.OpenKeyStream[ks];
    ButtonWait;
    SetCursor[textpointer];
  END;

DestroyWindow: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord]=
  BEGIN
  -- define locals
    i: INTEGER;
    doit: BOOLEAN;
    nw: WindowHandle;
    [nw, doit] ← WindowFrontEnd[w, bullseye];
    IF doit THEN
      BEGIN
      -- check if we can delete this one
      FOR i  IN [0..4) DO
        IF windows[i] = nw THEN
          BEGIN
          ButtonWait;
          SetCursor[textpointer];
          RETURN;
          END;
        ENDLOOP;
      -- check if one of our scratch files is in the window now
      FOR i IN [0..maxscratch) DO
        IF scratchfiles[i] = nw.file THEN
          -- deallocate it (gets automatically deleted!)
          BEGIN
          scratchfiles[i] ← NIL;
          EXIT;
          END;
        ENDLOOP;
      -- get rid of all stuff in the window
      IF nw.ds # NIL THEN nw.ds.destroy[nw.ds];
      IF nw.ks # NIL THEN nw.ks.destroy[nw.ks];
      IF nw.menu # NIL THEN DestroyMenu[nw.menu];
      DestroyRectangle[nw.rectangle];
      DestroyDisplayWindow[nw];
      -- repaint current if not deleted one
      IF nw # w THEN
        PaintDisplayWindow[w]
      ELSE-- destroy set a new guy current!!
        BEGIN
        nw ← GetCurrentDisplayWindow[];
        StreamDefs.OpenKeyStream[nw.ks];
        MarkSelection[nw];
        END;
    END;
    ButtonWait;
    SetCursor[textpointer];
  END;

MoveWindow: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord] =
  BEGIN
  -- define locals
    savex, mapx: xCoord;
    savey, mapy: yCoord;
    mb: AMouseButton;
    r: Rptr = w.rectangle;
  -- jam cursor
    x ← xmouseloc↑ ← xcursorloc↑ ← r.bitmap.x0 + r.x0;
    y ← ymouseloc↑ ← ycursorloc↑ ← r.bitmap.y0 + r.y0;
    SetCursor[leftbutton];
    [savex, savey] ← CursorToMapCoords[r.bitmap, x, y];
  -- while no buttons are down move it
    WHILE (mb ← GetMouseButton[]) # Red DO
         IF mb = Blue THEN
              BEGIN
              MoveRectangle[r,savex, savey];
              EXIT;
              END;
      [mapx, mapy] ← CursorToMapCoords[r.bitmap, x, y];
```

```
      x ← MAX[0, mapx];
      y ← MAX[0, mapy];
      MoveRectangle[r, x, y];
      x ← xcursorloc↑;
      y ← ycursorloc↑;
      ENDLOOP;
    ButtonWait;
  -- now paint it for possible cleanup
    PaintDisplayWindow[w];
    SetCursor[textpointer];
  END;

GrowWindow: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord]=
  BEGIN
  -- define locals
    savewidth, width: xCoord;
    saveheight, height: yCoord;
    mb: AMouseButton;
    r: Rptr = w.rectangle;
  -- first move the cursor to lower right corner
    x ← r.bitmap.x0+r.x0+r.cw;
    xmouseloc↑ ← xcursorloc↑ ← x;
    y ← r.bitmap.y0+r.y0+r.ch;
    ymouseloc↑ ← ycursorloc↑ ← y;
    SetCursor[leftbutton];
    [savewidth, saveheight] ←
                    CursorToRectangleCoords[w.rectangle, x, y];
  -- while no buttons are down grow it
    WHILE (mb ← GetMouseButton[]) # Red DO
      IF GetMouseButton[] = Blue THEN
              BEGIN
              GrowRectangle[r, savewidth, saveheight];
              EXIT;
              END;
      [width, height] ← CursorToRectangleCoords[w.rectangle, x, y];
      width ← MAX[width, 35];
      height ← MAX[height, 35];
      GrowRectangle[r, width, height];
      x ← xcursorloc↑; y ← ycursorloc↑;
      ENDLOOP;
    ButtonWait;
  -- now paint it for cleanup
    PaintDisplayWindow[w];
    SetCursor[textpointer];
  END;

LoadWindow: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord]=
  BEGIN
-- define locals
  i: INTEGER;
  doit: BOOLEAN;
  nw: WindowHandle;
  str, scratchstr: STRING;
  [nw, doit] ← WindowFrontEnd[w, leftbutton];
  SetCursor[hourglass];
  IF doit AND nw.type # scriptfile THEN
      BEGIN
-- check if one of our scratch files is in the window now
    FOR i IN [0..maxscratch) DO
      IF scratchfiles[i] = nw.file THEN
      -- deallocate it (gets automatically deleted!)
        BEGIN
        scratchfiles[i] ← NIL;
        EXIT;
        END;
      ENDLOOP;
    str ← GetSelection[w];
    IF str.length > 38 THEN
      str.length ← 38; -- max file name length
    AlterWindowType[nw, file, str
      ! SegmentDefs.FileNameError =>
        BEGIN
        NoteNameError[nw, str];
        CONTINUE;
        END
      ];
```

```
      SystemDefs.FreeHeapString[str];
      IF w = nw THEN PaintDisplayWindow[w];
      END;
    ButtonWait;
    SetCursor[textpointer];
    END;

FindSelection: PROCEDURE[w: WindowHandle, x: xCoord, y: yCoord]=
  BEGIN
  -- Declare Locals
  i: INTEGER;
  str: STRING;
  nw: WindowHandle;
  scrollto: StreamIndex;
  doit: BOOLEAN;
  noscroll: BOOLEAN;
  [nw, doit] ← WindowFrontEnd[w, leftbutton];
  SetCursor[hourglass];
  -- get current selection
  IF doit AND NOT EqualIndex[w.selection.rightindex, nullindex] THEN
     BEGIN
     str ← GetSelection[w];
     w ← GetCurrentDisplayWindow[];
     IF w # nw AND nw.file # NIL THEN
       OpenDiskStream[nw.file
         ! StreamError =>
           BEGIN
           nw.eofindex ← GetIndex[nw.file];
           RESUME
           END
           ];
     --do a text string search starting from the end of the current selection
     IF EqualIndex[nw.selection.rightindex, nullindex] THEN
       nw.selection.rightindex ← originindex;
     BEGIN
     [scrollto, nw.selection.leftindex, nw.selection.rightindex] ←
       TextSearch[nw, str, nw.selection.rightindex
         ! SearchFailed => GOTO fail];
     --scroll file to beginning of line containing text
     noscroll ← EqualIndex[scrollto, w.selection.rightindex];
     IF nw.type = scratch OR nw.type = scriptfile
       THEN nw.tempindex ← scrollto;
     IF NOT noscroll THEN SetIndexForWindow[nw, scrollto]
     ELSE IF nw = w THEN PaintDisplayWindow[nw];
     EXITS fail =>
       IF nw = w THEN PaintDisplayWindow[nw];
     END;
     IF nw # w AND nw.file # NIL
       THEN CloseDiskStream[nw.file];
     SystemDefs.FreeHeapString[str];
     END;
   ButtonWait;
   SetCursor[textpointer];
   END;

  SearchFailed: ERROR = CODE;

  TextSearch: PROCEDURE [w: WindowHandle, s: STRING, pos: StreamIndex] RETURNS [StreamIndex, StreamI
**ndex, StreamIndex] =
    -- searches for s using the Knuth, Pratt, Morris algorithm.
    -- returns stream index of the start of the line containing s.
    BEGIN
    i, j: INTEGER;
    char: CHARACTER;
    l: INTEGER = s.length;
    offset: INTEGER ← 1;
    ff: DESCRIPTOR FOR ARRAY OF INTEGER; -- failure function
    IF l = 0 THEN ERROR SearchFailed;  -- empty string
    ff ← DESCRIPTOR[SystemDefs.AllocateHeapNode[l],l];
    -- set up failure function
    j ← 0; i ← ff[0] ← -1;
    WHILE j < l-1 DO
      WHILE i >= 0 AND s[j] # s[i] DO i ← ff[i] ENDLOOP;
      i ← i+1; j ← j+1;
      ff[j] ← IF s[j] = s[i] THEN ff[i] ELSE i;
      ENDLOOP;
```

```
    SetIndex[w.file,pos];
    char ← w.file.get[w.file];
    j ← 0;  -- j = pattern index
    DO ENABLE UNWIND => SystemDefs.FreeHeapNode[BASE[ff]];
      IF j >= 1 THEN EXIT;
      char ← w.file.get[w.file ! StreamError => GOTO fail];
      offset ← offset+1;
      IF char = CR THEN
        BEGIN pos ← ModifyIndex[pos,offset]; offset ← 0 END;
      WHILE j >= 0 AND char # s[j] DO j ← ff[j] ENDLOOP;
      j ← j + 1;
      REPEAT
        fail => ERROR SearchFailed;
      ENDLOOP;
    SystemDefs.FreeHeapNode[BASE[ff]];
    RETURN[pos, ModifyIndex[pos,offset-1], ModifyIndex[pos,offset-1]]
    END;

WindowFrontEnd: PUBLIC PROCEDURE[w:  WindowHandle, cursor: CursorType]
   RETURNS[WindowHandle, BOOLEAN]=
  BEGIN
  -- Declare Locals
  nw:  WindowHandle;
  x:  xCoord; y:  yCoord;
  mb:  AMouseButton;
  doit:  BOOLEAN ← TRUE;
  -- ask for window to put stuff into
  SetCursor[cursor];
  -- wait to select a window or say ignore
  UNTIL (mb ← GetMouseButton[]) = Red DO
    IF mb = Blue THEN
      BEGIN
      doit ← FALSE;
      RETURN[w, doit]
      END;
     ENDLOOP;
   -- now get selected window
   x ← xcursorloc↑; y ← ycursorloc↑;
   [nw, x, y] ← FindDisplayWindow[x, y];
   IF nw = NIL THEN nw ← w;
   RETURN[nw, doit];
  END;

  KeysetMessage: ARRAY BOOLEAN OF STRING ← ["KEYS ON", "KEYS OFF"];

  EnableKeyset: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord] =
    BEGIN
    -- define locals
    menuarray[commands-1].keyword ← KeysetMessage[useKeyset ← ~useKeyset];
    END;



ButtonWait:  PROCEDURE =
  BEGIN
  -- wait until all button are up
  UNTIL GetMouseButton[] = None DO
    NULL;
    ENDLOOP;
  RETURN;
  END;

-- initialization for windows module
InitWindows:  PROCEDURE =
  BEGIN
  menuarray ←
    [
    MenuItem[" CREATE", CreateWindow],
    MenuItem["DESTROY", DestroyWindow],
    MenuItem["  MOVE", MoveWindow],
    MenuItem["  GROW", GrowWindow],
    MenuItem["  LOAD", LoadWindow],
    MenuItem["STUFF IT", PutSelect],
    MenuItem["  FIND", FindSelection],
    MenuItem[KeysetMessage[useKeyset ← FALSE], EnableKeyset]
    ];
```

```
  END;

-- MAIN BODY CODE
InitWindows[];


END. of wmanwindows
```