```
--File:  WManPosition.mesa
--Edited by Sandman      October 7, 1977  12:34 PM

DIRECTORY
  AltoDefs:  FROM "altodefs",
  DoubleDefs: FROM "doubledefs",
  InlineDefs:  FROM "inlinedefs",
  StreamDefs:  FROM "streamdefs",
  MenuDefs:  FROM "menudefs",
  RectangleDefs:  FROM "rectangledefs",
  WindowDefs:  FROM "windowdefs",
  WManagerDefs:  FROM "wmanagerdefs";

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

WManPosition:  PROGRAM[WMState:  WMDataHandle]
  IMPORTS DoubleDefs, StreamDefs, RectangleDefs, WindowDefs, WManagerDefs
  EXPORTS WManagerDefs
  SHARES StreamDefs, WManagerDefs =
  BEGIN
  OPEN WMState;

  CR: CHARACTER = 15C;
  Space: CHARACTER = 40C;

  PositionFile: PUBLIC PROCEDURE[w: WindowHandle, x: xCoord, y: yCoord]=
    BEGIN OPEN DoubleDefs;
    -- Declare Locals
    height: CARDINAL;
    bytepos, eof: LongCARDINAL;
    index: StreamIndex;
    -- compute position in file and set it
    SetCursor[arrow];
    ButtonWait;
    SetCursor[hourglass];
    x ← xcursorloc↑; y ← ycursorloc↑;
    [x, y] ← CursorToRectangleCoords[w.rectangle, x, y];
    -- if out of jump bar then no scrolling
    IF NOT CheckForSlop[w, x, y] THEN
      BEGIN
      SetJumpStripe[w, FALSE];
      RETURN;
      END;
    IF y < defaultlineheight+1 OR w.eofindex.byte = 177777B THEN index ← [0, 0]
    ELSE
      BEGIN OPEN InlineDefs, AltoDefs;
      height ← w.rectangle.ch-(defaultlineheight+1);
      y ← MIN[LOOPHOLE[y-(defaultlineheight+1), CARDINAL], height];
      IF y = height THEN index ← w.eofindex
      ELSE
        BEGIN
        eof ← DAdd[LongMult[w.eofindex.page, BytesPerPage],
          LongCARDINAL[w.eofindex.byte, 0]];
        bytepos ← DDivide[DMultiply[eof, LongCARDINAL[y, 0]],
          LongCARDINAL[height, 0]].quotient;
        [index.page, index.byte] ← LongDivMod[bytepos, BytesPerPage];
        IF index.page > w.eofindex.page OR (index.page = w.eofindex.page
          AND index.byte > w.eofindex.byte) THEN index ← w.eofindex;
        END;
      END;
    DoTheScroll[w, index];
    END;

  ScrollUpFile: PUBLIC PROCEDURE[w: WindowHandle, x: xCoord, y: yCoord]=
    BEGIN
    -- Declare Locals
    index: StreamIndex;
    line:  INTEGER;
    -- compute position in file and set it
    SetCursor[uparrow];
    ButtonWait;
    SetCursor[hourglass];
    x ← xcursorloc↑; y ← ycursorloc↑;
    [line, , ,index] ← ResolveBugToPosition[w, x, y];
    [x, y] ← CursorToRectangleCoords[w.rectangle, x, y];
    -- if out of jump bar then no scrolling
    IF NOT CheckForSlop[w, x, y] OR line = 1 THEN
      BEGIN
      SetJumpStripe[w, FALSE];
      RETURN;
      END;
    DoTheScroll[w, index];
    END;

  ScrollDownFile: PUBLIC PROCEDURE[w: WindowHandle, x: xCoord, y: yCoord]=
    BEGIN OPEN DoubleDefs, InlineDefs;
    -- Declare Locals
    index, posindex: StreamIndex;
    maxbackup,pos: LongCARDINAL;
    line,nlines:  CARDINAL;
    nlines ← (w.rectangle.ch/w.ds.lineheight)-1;
    -- compute position in file and set it
    SetCursor[downarrow];
    ButtonWait;
    SetCursor[hourglass];
    x ← xcursorloc↑; y ← ycursorloc↑;
    [x, y] ← CursorToRectangleCoords[w.rectangle, x, y];
    line ← MIN[LOOPHOLE[MAX[1, y/w.ds.lineheight],CARDINAL], nlines];
    posindex ← SELECT w.type FROM
      scratch, scriptfile =>
        IF w.tempindex = nullindex THEN w.fileindex ELSE w.tempindex,
      file => w.fileindex,
      ENDCASE => originindex;
    pos ← DAdd[LongMult[posindex.page, AltoDefs.BytesPerPage], [posindex.byte, 0]];
    -- if out of jump bar or first window then nop
    IF NOT CheckForSlop[w, x, y] OR EqualIndex[posindex, originindex] THEN
      BEGIN
      SetJumpStripe[w, FALSE];
      RETURN;
      END;
    maxbackup ← LongMult[w.rectangle.cw/ComputeCharWidth[Space,w.ds.pfont], line];
    IF DCompare[pos, maxbackup] = Comparison[greater] THEN
      BEGIN
      maxbackup ← DSub[pos, maxbackup];
      [index.page, index.byte] ← LongDivMod[maxbackup, AltoDefs.BytesPerPage];
      END
    ELSE index ← originindex;
    index ← GenerateLineTable[w,index,posindex,line,nlines];
    DoTheScroll[w, index];
    END;

  NormalizeSelection: PUBLIC PROCEDURE[w: WindowHandle, x: xCoord, y: yCoord]=
    BEGIN OPEN DoubleDefs, InlineDefs;
    --Declare locals
    linestarts:  DESCRIPTOR FOR ARRAY OF StreamIndex;
    maxbackup,pos: LongCARDINAL;
    index: StreamIndex;
    line:  INTEGER;
    nlines:  CARDINAL;
    nlines ← (w.rectangle.ch/w.ds.lineheight)-1;
    linestarts ← DESCRIPTOR[GetLineTable[], nlines+1];
    -- compute position in file and set it
    SetCursor[norm];
    ButtonWait;
    SetCursor[hourglass];
    x ← xcursorloc↑; y ← ycursorloc↑;
    [x, y] ← CursorToRectangleCoords[w.rectangle, x, y];
    line ← MIN[MAX[1, y/w.ds.lineheight], nlines];
    -- if out of jump bar then nop
    IF NOT CheckForSlop[w, x, y] THEN
      BEGIN
      SetJumpStripe[w, FALSE];
      RETURN;
      END;
    --if no selection or no scroll, simply move to beginning of file
    IF EqualIndex[w.selection.leftindex, nullindex]
     OR (EqualIndex[linestarts[0], originindex]
     AND line > w.selection.leftline)
      THEN index ← originindex
    -- selection visible and below bug
    ELSE IF w.selection.leftline # 0 AND
     (GrEqualIndex[w.selection.leftindex, linestarts[line-1]]
     OR line <= 2 * w.selection.leftline)
      THEN index ← linestarts[ABS[w.selection.leftline - line]]
    -- adjustments necessary
    ELSE BEGIN
      pos ← DAdd[LongMult[w.selection.leftindex.page, AltoDefs.BytesPerPage],
        [w.selection.leftindex.byte, 0]];
      maxbackup ← LongMult[w.rectangle.cw/ComputeCharWidth[Space,w.ds.pfont], line];
      IF DCompare[pos, maxbackup] = Comparison[greater] THEN
        BEGIN
        maxbackup ← DSub[pos, maxbackup];
        [index.page, index.byte] ← LongDivMod[maxbackup, AltoDefs.BytesPerPage];
        END
      ELSE index ← originindex;
      -- get within window range
      index ← GenerateLineTable[w,index,w.selection.leftindex,line,nlines];
      END;
    DoTheScroll[w, index];
    END;

  CheckForSlop:  PROCEDURE[w:  WindowHandle, x:  xCoord, y:  yCoord]
    RETURNS[BOOLEAN]=
    BEGIN
    flag:  BOOLEAN  ← FALSE;
    --check if some part of cursor is in jump bar
    IF (x+slop > 0 AND x <= JumpStrip + 15 AND y+slop > 0
      AND y - slop <= w.rectangle.ch)
      THEN flag ← TRUE;
    RETURN[flag];
    END;

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

  DoTheScroll:  PROCEDURE[w:  WindowHandle, index:  StreamIndex]=
    BEGIN
    SELECT w.type FROM
      clear => NULL;
      random => NULL;
      scratch,
      scriptfile =>
        BEGIN
        IF index = w.tempindex THEN RETURN;
        w.tempindex ← index;
        w.ds.options.StopBottom ← TRUE;
        IF w = GetCurrentDisplayWindow[] THEN
          BEGIN
          PaintDisplayWindow[w];
          END;
        END;
      file =>
        BEGIN
        IF index = w.fileindex THEN RETURN;
        w.fileindex ← index;
        IF w = GetCurrentDisplayWindow[] THEN
          BEGIN
          PaintDisplayWindow[w];
          END;
        END;
     ENDCASE;
   -- say not in jump mode anymore
   SetJumpStripe[w, FALSE];
   END;

  GenerateLineTable:  PROCEDURE [w: WindowHandle, topindex, find: StreamIndex,
   line, big: CARDINAL] RETURNS [StreamIndex]  =
    BEGIN
    -- declare locals
    ptr:  ARRAY[0..maxlines) OF StreamIndex;
    i, x:  CARDINAL;
    char:  CHARACTER;
    once:  BOOLEAN ← TRUE;
    index, savedindex:  StreamIndex;
    x ← leftmargin;
    savedindex ← GetIndex[w.file];
    SetIndex[w.file, topindex];
    index ← topindex;
    FOR i IN [0..big) DO
      ptr[i] ← nullindex;
      ENDLOOP;
    i ← 0;
    -- generate the table
    WHILE NOT EqualIndex[index, find] DO
      index ← GetIndex[w.file];
      char ← w.file.get[w.file];
      x ← x + ComputeCharWidth[char,w.ds.pfont];
      IF x >= w.rectangle.cw OR char = CR THEN
        BEGIN
        x ← leftmargin;
        IF char = CR THEN index ← GetIndex[w.file];
        ptr[i] ← index;
        i ← (i + 1) MOD big;
        END;
      ENDLOOP;
    index ← ptr[LOOPHOLE[big-line+i, CARDINAL] MOD big];
    IF NOT EqualIndex[index, nullindex] THEN topindex ← index;
    SetIndex[w.file,savedindex];
    RETURN[topindex];
    END;

  -- initialization for position module

  InitPosition:  PROCEDURE =
    BEGIN
    ScrollProcArray[RedYellowBlue] ← NullProc;
    ScrollProcArray[RedBlue] ← NormalizeSelection;
    ScrollProcArray[RedYellow] ← NullProc;
    ScrollProcArr^y[Red] ← ScrollUpFile;
    ScrollProcArray[BlueYellow] ← NullProc;
    ScrollProcArray[Blue] ← ScrollDownFile;
    ScrollProcArray[Yellow] ← PositionFile;
    ScrollProcArray[None] ← NullProc;
    END;

  --MAIN BODY CODE

  InitPosition[];


  END. of wmanposition
```