```
--File:  WEPosition.mesa
--Edited by:
--           Sandman  March 29, 1978  2:02 PM
--           Barbara  July 25, 1978  9:52 AM

DIRECTORY
  AltoDefs: FROM "altodefs" USING [BytesPerPage],
  RectangleDefs: FROM "rectangledefs" USING [ComputeCharWidth, leftmargin],
  StreamDefs: FROM "streamdefs" USING [
    EqualIndex, GetIndex, GrEqualIndex, SetIndex],
  WindExDefs: FROM "windexdefs" USING [
    CursorToRectangleCoords, GetMouseButton, JumpStrip, LDivMod, LMult,
    maxlines, NullIndex, NullProc, OriginIndex, SetCursor, SetJumpStripe,
    slop, WEDataHandle,xcursorloc, ycursorloc],
  WindowDefs: FROM "windowdefs" USING [
    GetCurrentDisplayWindow, GetLineTable, PaintDisplayWindow,
    ResolveBugToPosition, StreamIndex, WindowHandle, xCoord, yCoord];

DEFINITIONS FROM StreamDefs, RectangleDefs, WindowDefs, WindExDefs;

WEPosition: PROGRAM [WEState:  WEDataHandle]
  IMPORTS StreamDefs, RectangleDefs, WindowDefs, WindExDefs
  EXPORTS WindExDefs
  SHARES StreamDefs, WindExDefs =
  BEGIN
  OPEN WEState;

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

  PositionFile: PUBLIC PROCEDURE[w: WindowHandle, x: xCoord, y: yCoord]=
    BEGIN
    -- Declare Locals
    height: CARDINAL;
    bytepos, eof: LONG INTEGER;
    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 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 ← LMult[w.eofindex.page, BytesPerPage] + w.eofindex.byte;
        bytepos ← (eof*y)/height;
        [index.page, index.byte] ← LDivMod[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[];
    x ← xcursorloc↑+cxa; y ← ycursorloc↑+cxa;
    [line, , ,index] ← ResolveBugToPosition[w, x, y];
```

```
    [x, y] ← CursorToRectangleCoords[w.rectangle, x-cxa, y-cya];
    SetCursor[hourglass];
    -- 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
    -- Declare Locals
    index, posindex: StreamIndex;
    maxbackup, pos: LONG INTEGER;
    line,nlines:  CARDINAL;
    nlines ← (w.rectangle.ch/w.ds.lineheight)-1;
    -- compute position in file and set it
    SetCursor[downarrow];
    ButtonWait[];
    x ← xcursorloc↑; y ← ycursorloc↑;
    [x, y] ← CursorToRectangleCoords[w.rectangle, x, y];
    SetCursor[hourglass];
    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 ← LMult[posindex.page, AltoDefs.BytesPerPage] + posindex.byte;
    -- 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 ← LMult[w.rectangle.cw/ComputeCharWidth[Space,w.ds.pfont], line];
    IF pos > maxbackup THEN
      BEGIN
      maxbackup ← pos-maxbackup;
      [index.page, index.byte] ← LDivMod[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
    linestarts: DESCRIPTOR FOR ARRAY OF StreamIndex;
    maxbackup, pos: LONG INTEGER;
    index: StreamIndex;
    lastindex: StreamIndex ← NullIndex;
    line, nlines, i:  CARDINAL;
    nlines ← (w.rectangle.ch/w.ds.lineheight)-1;
    linestarts ← DESCRIPTOR[GetLineTable[], nlines+1];
    -- compute position in file and set it
    SetCursor[norm];
    ButtonWait[];
    x ← xcursorloc↑; y ← ycursorloc↑;
    [x, y] ← CursorToRectangleCoords[w.rectangle, x, y];
    SetCursor[hourglass];
    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;
    FOR i IN [1..nlines) DO
      IF EqualIndex[NullIndex, linestarts[i]] THEN
        BEGIN lastindex ← w.eofindex; EXIT; END;
      REPEAT
        FINISHED => lastindex ← linestarts[nlines];
      ENDLOOP;
```

```
    --if no selection or no scroll, simply move to beginning of file
    IF EqualIndex[w.selection.leftindex, NullIndex] OR
     --selection past current end-of-file
     GrEqualIndex[w.selection.leftindex, lastindex] 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[IF w.selection.leftline >= line
        THEN w.selection.leftline - line ELSE line - w.selection.leftline]
    -- adjustments necessary
    ELSE BEGIN
      pos ← LMult[w.selection.leftindex.page, AltoDefs.BytesPerPage] +
        w.selection.leftindex.byte;
      maxbackup ← LMult[w.rectangle.cw/ComputeCharWidth[Space,w.ds.pfont], line];
      IF pos > maxbackup THEN
        BEGIN
        maxbackup ← pos - maxbackup;
        [index.page, index.byte] ← LDivMod[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] ← NullProc;
  ScrollProcArray[RedYellow] ← NullProc;
  ScrollProcArray[Red] ← ScrollUpFile;
  ScrollProcArray[BlueYellow] ← NormalizeSelection;
  ScrollProcArray[Blue] ← ScrollDownFile;
  ScrollProcArray[Yellow] ← PositionFile;
  ScrollProcArray[None] ← NullProc;
  END;

--MAIN BODY CODE

InitPosition[];


END. of weposition
```