```
-- SystemDisplay.mesa; edited by Johnsson; October 14, 1977  4:17 PM

DIRECTORY
  AltoDefs: FROM "altodefs",
  DisplayDefs: FROM "displaydefs",
  FontDefs: FROM "fontdefs",
  ImageDefs: FROM "imagedefs",
  InlineDefs: FROM "inlinedefs",
  SegmentDefs: FROM "segmentdefs",
  StreamDefs: FROM "streamdefs";

SystemDisplay: PROGRAM
  IMPORTS ImageDefs, SegmentDefs, StreamDefs
  EXPORTS DisplayDefs, StreamDefs
  SHARES StreamDefs =

  BEGIN

  StreamHandle: TYPE = StreamDefs.StreamHandle;
  OrderedPOINTER: TYPE = ORDERED POINTER;
  OrderedNIL: OrderedPOINTER = LOOPHOLE[NIL];

  TAB: CHARACTER = 11C;
  CR: CHARACTER = 15C;
  NUL: CHARACTER = 0C;
  SP: CHARACTER = ' ;

-- Display Hardware

  DCBchainHead: DCBptr = LOOPHOLE[420B];
  DCBnil: DCBptr = LOOPHOLE[0];
  DCBptr: TYPE = POINTER TO DCB;
  DCB: TYPE = RECORD [
    next: DCBptr,
    resolution: {high,low},
    background: DisplayDefs.Background,
    indenting: [0..77B],                -- in units of 16 bits
    width: [0..377B],           -- in units of 16 bits, must be even
    bitmap: OrderedPOINTER,             -- must be even
    height: CARDINAL];          -- in double scan lines
  LeftMargin: CARDINAL = 8;
  RightMargin: CARDINAL = 606;
  MaxWordsPerLine: CARDINAL = 38;

  DSP: Display StreamDefs.StreamObject ← StreamDefs.StreamObject [
    reset: ClearDS,
    get: GetNop,
    put: DPutChar,
    putback: PutbackNop,
    endof: EndofNop,
    destroy: DestroyNop,
    body: Display[,,,,,,,,,]];

  systemDS: StreamDefs.DisplayHandle = @DSP;

  displayOn: BOOLEAN ← FALSE;
  bmSegment: SegmentDefs.DataSegmentHandle;
  bmFirst, bmTail, bmNext, bmLastLine: OrderedPOINTER;
  lineHeight, lastLineSize: CARDINAL;
  dummyDCB: DCBptr;
  firstDCB, lastDCB, currentDCB: DCBptr ← DCBnil;
  -- layout of bitmap is:
  --    DCBs: ARRAY [firstDCB..lastDCB] OF DCB,
  --    bmFirst: ARRAY OF UNSPECIFIED,
  --    bmLastLine: ARRAY [0..lineHeight*MaxWordsPerLine) OF UNSPECIFIED,
  -- bmNext points to next word to allocate,
  -- bmTail points to oldest allocated bitmap.

  bmState: FontDefs.BitmapState;
  tabWidth: CARDINAL;
  TABindex: [0..9];
  TABs: ARRAY [0..9] OF CARDINAL;

  font: FontDefs.FontHandle ← NIL;

  -- Typescript data
```

```
  typescript: PUBLIC StreamDefs.DiskHandle ← NIL;
  startLine: StreamDefs.StreamIndex ← [0, 0];

  GetDefaultDisplayStream: PUBLIC PROCEDURE RETURNS[StreamDefs.DisplayHandle] =
    BEGIN
    RETURN[systemDS];
    END;

  SetupBitmap: PROCEDURE [bitmap: POINTER, nLines, nWords: CARDINAL] =
    BEGIN
    dcb: DCBptr;
    WHILE nLines*SIZE[DCB] + lastLineSize > nWords DO
      nLines ← nLines - 1;
      ENDLOOP;
    firstDCB ← dcb ← bitmap;
    THROUGH [0..nLines) DO
      dcb.next ← dcb + SIZE[DCB];
      dcb.height ← lineHeight/2;
      dcb ← dcb.next;
      ENDLOOP;
    lastDCB ← dcb-SIZE[DCB];
    lastDCB.next ← DCBnil;
    bmFirst ← LOOPHOLE[dcb, OrderedPOINTER];
    bmLastLine ← LOOPHOLE[bitmap+nWords-lastLineSize, OrderedPOINTER];
    bmState ← [origin: bmLastLine, wordsPerLine: MaxWordsPerLine, x:, y:0];
    END;

  currentPages, currentLines, currentDummySize: CARDINAL;

  DisplayOff: PUBLIC PROCEDURE [color: DisplayDefs.Background] =
    BEGIN
    IF ~displayOn THEN RETURN;
    SetSystemDisplaySize[0,0];
    font.close[font];
    dummyDCB.background ← color;
    dummyDCB.height ← 1;
    END;

  DisplayOn: PUBLIC PROCEDURE =
    BEGIN
    IF displayOn THEN RETURN;
    dummyDCB.background ← white;
    SetDummyDisplaySize[currentDummySize];
    SetSystemDisplaySize[currentLines, currentPages];
    END;

  SetSystemDisplaySize: PUBLIC PROCEDURE [nTextLines, nPages: CARDINAL] =
    BEGIN OPEN SegmentDefs;
    IF displayOn THEN
      BEGIN
      firstDCB.next ← lastDCB.next;
      RemoveDCB[firstDCB];
      firstDCB ← lastDCB ← currentDCB ← DCBnil;
      DeleteDataSegment[bmSegment];
      displayOn ← FALSE;
      END;
    IF nPages = 0 THEN RETURN;
    currentPages ← nPages;  -- for Display Off/On
    currentLines ← nTextLines;  -- for Display Off/On
    bmSegment ← NewDataSegment[DefaultBase, nPages];
    SetupBitmap[DataSegmentAddress[bmSegment], nTextLines, nPages*AltoDefs.PageSize];
    displayOn ← TRUE;
    ClearDS[systemDS];
    InsertDCB[new: firstDCB, before: dummyDCB.next];
    RETURN
    END;

  SetDummyDisplaySize: PUBLIC PROCEDURE [nScanLines: CARDINAL] =
    BEGIN
    currentDummySize ← nScanLines;  -- for Display Off/On
    IF nScanLines/2 = dummyDCB.height THEN RETURN;
    IF dummyDCB.height # 0 THEN RemoveDCB[dummyDCB];
    dummyDCB.height ← nScanLines/2;
    IF dummyDCB.height # 0 THEN InsertDCB[new: dummyDCB, before: firstDCB];
    RETURN
    END;
```

```
ClearDS: PROCEDURE [stream: StreamHandle] =
  BEGIN
  dcb: DCBptr;
  IF stream # systemDS THEN
    SIGNAL StreamDefs.StreamError[stream,StreamType];
  IF typescript # NIL THEN typescript.reset[typescript];
  IF ~displayOn THEN RETURN;
  FOR dcb ← firstDCB, dcb.next DO
    dcb.resolution ← high;
    dcb.background ← white;
    dcb.indenting ← dcb.width ← 0;
    dcb.bitmap ← bmLastLine;
    IF dcb = lastDCB THEN EXIT;
    ENDLOOP;
  bmNext ← bmFirst;
  bmTail ← bmLastLine;
  currentDCB ← firstDCB;
  ClearCurrentLine[stream];
  RETURN
  END;


ClearCurrentLine: PUBLIC PROCEDURE [stream: StreamHandle] =
  BEGIN
  IF stream # systemDS THEN
    SIGNAL StreamDefs.StreamError[stream,StreamType];
  IF typescript # NIL THEN
    StreamDefs.SetIndex[typescript, startLine];
  IF ~displayOn THEN RETURN;
  bmLastLine↑ ← 0;
  InlineDefs.COPY[from: bmLastLine, to: bmLastLine+1, nwords: lastLineSize-1];
  currentDCB.indenting ← 0;
  currentDCB.bitmap ← bmLastLine;
  currentDCB.width ← MaxWordsPerLine;
  bmState.x ← LeftMargin;
  TABindex ← 0;
  RETURN
  END;


Scroll: PROCEDURE [char: CHARACTER] =
  BEGIN
  dcb: DCBptr;
  pos: CARDINAL;
  SELECT char FROM
    CR => NULL;
    TAB =>
      BEGIN
      TABs[TABindex] ← bmState.x;
      TABindex ← TABindex + 1;
      pos ← (bmState.x/tabWidth+1)*tabWidth;
      IF pos < RightMargin THEN bmState.x ← pos
      ELSE DPutChar[systemDS, SP];
      RETURN
      END;
    NUL => RETURN;
    ENDCASE =>
      IF char < 40C THEN
        BEGIN
        DPutChar[systemDS, '↑];
        DPutChar[systemDS,
          LOOPHOLE[LOOPHOLE[char,CARDINAL]+100B,CHARACTER]];
        RETURN
        END;
  -- Do the scroll, assuming last (current) line is in bmlastline.
  -- scroll all others by BITing their DCBs.  move old last line to
  -- new bitmap and free bmlastLine for reuse.
  UNTIL Compact[currentDCB,bmState.x] DO
    IF ~DeleteTopline[] THEN RETURN;-- not enough space
    ENDLOOP;
  IF currentDCB # lastDCB THEN currentDCB ← currentDCB.next
  ELSE
    BEGIN
    IF firstDCB.width # 0 THEN
      [] ← DeleteTopline[];
```

```
    FOR dcb ← firstDCB, dcb.next UNTIL dcb = lastDCB DO
      InlineDefs.COPY[from: dcb.next+1, to: dcb+1, nwords: SIZE[DCB]-1];
      -- assumes "next" is in word zero
      ENDLOOP;
    END;
  IF typescript # NIL THEN startLine ← StreamDefs.GetIndex[typescript];
  ClearCurrentLine[systemDS];
  IF char # CR THEN DPutChar[systemDS, char];
  END;


DeleteTopLine: PROCEDURE RETURNS [BOOLEAN] =
  BEGIN
  dcb: DCBptr;
  -- find first line with bitmap allocated
  FOR dcb ← firstDCB, dcb.next DO
    IF dcb.width # 0 THEN EXIT;
    IF dcb = lastDCB THEN RETURN[FALSE]; -- found no top line to delete
    ENDLOOP;
  dcb.width ← dcb.indenting ← 0;
  -- find next line with bitmap allocated
  UNTIL dcb = lastDCB DO
    dcb ← dcb.next;
    IF dcb.width # 0 THEN
      BEGIN bmTail ← dcb.bitmap; EXIT END;
    REPEAT FINISHED => -- all linex deleted
      BEGIN bmTail ← bmLastLine; bmNext ← bmFirst END;
    ENDLOOP;
  RETURN[TRUE];
  END;


Compact: PROCEDURE [dcb: DCBptr, x: CARDINAL] RETURNS [BOOLEAN] =
  BEGIN
  newWidth: CARDINAL ← (x+15)/16;
  oldWidth: CARDINAL ← dcb.width;
  old: OrderedPOINTER ← dcb.bitmap;
  lineHeight: CARDINAL ← dcb.height*2;
  p, new: OrderedPOINTER;
  d: CARDINAL;
  IF x <= LeftMargin THEN d ← 0
  ELSE
    FOR d IN [0..newWidth) DO
      p ← old + d;
      THROUGH [0..lineHeight) DO
        IF p↑ # 0 THEN GO TO foundit;
        p ← p + oldWidth;
        ENDLOOP;
      REPEAT foundit => NULL;
      ENDLOOP;
  IF d > 0 THEN
    BEGIN
    newWidth ← newWidth-d;
    old ← old + d;
    END;
  newWidth ← Even[newWidth];
  IF newWidth > 0 THEN
    BEGIN
    IF (p ← new ← GetMapSpace[newWidth*lineHeight]) = OrderedNIL THEN RETURN[FALSE];
    THROUGH [0..lineHeight) DO
      InlineDefs.COPY[from: old, to: p, nwords: newWidth];
      old ← old + oldWidth;
      p ← p + newWidth;
      ENDLOOP;
    dcb.indenting ← d;
    dcb.width ← newWidth;
    dcb.bitmap ← new;
    END
  ELSE dcb.indenting ← dcb.width ← 0;
  RETURN [ TRUE ];
  END;

GetMapSpace: PROCEDURE [nwords: [0..77777B]] RETURNS [p: OrderedPOINTER] =
  BEGIN
  t: INTEGER;
  DO
```

```
      t ← bmTail - bmNext;
      IF t < 0 THEN
        IF bmLastLine >= bmNext+nwords THEN EXIT
        ELSE bmNext ← bmFirst
      ELSE
        IF t >= nwords THEN EXIT
        ELSE RETURN[OrderedNIL];
      ENDLOOP;
    p ← bmNext;
    bmNext ← bmNext + nwords;
    RETURN
    END;

  DPutChar: PROCEDURE [stream: StreamHandle, char: UNSPECIFIED] =
    BEGIN
    IF stream # systemDS THEN SIGNAL StreamDefs.StreamError[stream,StreamType];
    IF ~displayOn THEN RETURN;
    IF char > 377B THEN RETURN;
    IF char < 40B OR
       bmState.x + font.charWidth[font, char] > RightMargin THEN Scroll[char]
    ELSE font.paintChar[font,char,@bmState];
    RETURN
    END;

  DPutCharTS: PROCEDURE [stream: StreamHandle, char: UNSPECIFIED] =
    BEGIN
    DPutChar[stream, char];
    IF typescript # NIL THEN typescript.put[typescript, char];
    RETURN
    END;

  ClearChar: PROCEDURE [stream: StreamHandle, char: UNSPECIFIED] =
    BEGIN
    IF stream # systemDS THEN SIGNAL StreamDefs.StreamError[stream,StreamType];
    IF displayOn THEN
      BEGIN
      SELECT char FROM
        NUL, CR, > 377B => RETURN;
        TAB =>
          BEGIN
          IF TABindex > 0 THEN
            BEGIN
            TABindex ← TABindex -1;
            bmState.x ← TABs[TABindex];
            END;
          RETURN
          END;
        < 40B =>
          BEGIN
          ClearDisplayChar[stream, char+100B];
          char ← '↑;
          END;
        ENDCASE => NULL;
      font.clearChar[font,char,@bmState];
      END;
    RETURN
    END;

  ClearDisplayChar: PUBLIC PROCEDURE [stream: StreamHandle, char: UNSPECIFIED] =
    BEGIN
    ClearChar[stream, char];
    IF typescript # NIL THEN
      BEGIN OPEN StreamDefs;
      SetIndex[typescript, ModifyIndex[GetIndex[typescript],-1]];
      END;
    RETURN
    END;

  GetNop: PROCEDURE [stream: StreamHandle] RETURNS [UNSPECIFIED] =
    BEGIN
    ERROR StreamDefs.StreamError[stream,StreamAccess]
    END;

  PutbackNop: PROCEDURE [stream: StreamHandle, char: UNSPECIFIED] =
    BEGIN
    ERROR StreamDefs.StreamError[stream,StreamAccess]
```

```
    END;

  EndofNop: PROCEDURE [stream: StreamHandle] RETURNS [BOOLEAN] =
    BEGIN
    IF stream # systemDS THEN SIGNAL StreamDefs.StreamError[stream,StreamType];
    RETURN[FALSE]
    END;

  DestroyNop: PROCEDURE [stream: StreamHandle] =
    BEGIN
    ERROR StreamDefs.StreamError[stream,StreamAccess]
    END;

  ddarray: ARRAY [0..SIZE[DCB]] OF UNSPECIFIED;

  InsertDCB: PROCEDURE [new: DCBptr, before: DCBptr] =
    BEGIN
    dcb: DCBptr;
    FOR dcb ← new, dcb.next DO
      IF dcb.next = DCBnil THEN
        BEGIN dcb.next ← before; EXIT END;
      ENDLOOP;
    FOR dcb ← DCBchainHead, dcb.next DO
      IF dcb.next = before THEN
        BEGIN dcb.next ← new; EXIT END;
      ENDLOOP;
    END;

  RemoveDCB: PROCEDURE [dcb: DCBptr] =
    BEGIN
    prev: DCBptr;
    FOR prev ← DCBchainHead, prev.next UNTIL prev.next = DCBnil DO
      IF prev.next = dcb THEN
        BEGIN prev.next ← dcb.next; EXIT END;
      ENDLOOP;
    dcb.next ← DCBnil;
    END;

  Even: PROCEDURE [a: UNSPECIFIED] RETURNS [UNSPECIFIED] =
    BEGIN RETURN[a + LOOPHOLE[a,CARDINAL] MOD 2] END;

  SetFont: PUBLIC PROCEDURE [f: FontDefs.FontHandle] =
    BEGIN OPEN SegmentDefs;
    font ← f;
    lineHeight ← Even[font.charHeight[font,'A]];
    lastLineSize ← lineHeight*MaxWordsPerLine;
    tabWidth ← font.charWidth[font, SP]*8;
    RETURN
    END;

  SetTypeScript: PUBLIC PROCEDURE [ts: StreamDefs.DiskHandle] =
    BEGIN
    systemDS.put ← IF (typescript ← ts) = NIL THEN DPutChar ELSE DPutCharTS;
    RETURN
    END;

  InitDisplay: PUBLIC PROCEDURE [dummySize, textLines, nPages: CARDINAL, f: FontDefs.FontHandle] =
    BEGIN OPEN SegmentDefs;
    IF font # NIL THEN font.destroy[font];
    SetFont[f];
    SetDummyDisplaySize[dummySize];
    SetSystemDisplaySize[textLines, nPages];
    RETURN
    END;

  CleanupDisplay: ImageDefs.CleanupItem ← [link:,proc:Cleanup];

  Cleanup: ImageDefs.CleanupProcedure =
    BEGIN
    SELECT why FROM
      Finish, Abort => DCBchainHead.next ← DCBnil;
      ENDCASE;
    END;

  ImageDefs.AddCleanupProcedure[@CleanupDisplay];
  dummyDCB ← Even[BASE[ddarray]];
```

```
  dummyDCB↑ ← [DCBnil,high,white,0,0,LOOPHOLE[0,OrderedPOINTER],0];
  END.
```