

File created: 16-Mar-2022 08:06:56 {DSK}<Users>kaplan>Local>medley3.5>my-medley>lispusers>ACE>ACE.;2

changes to: (VARS ACECOMS ACE-MAINCOMS ACE-PRIMCOMS ACE.PIXPERWORD ACE.BITMAP.MASK ACE-EDITCOMS)
(RECORDS ACE.FRAME ACE.BLIT)
(MACROS ACE.MT.SCRX.SEQX ACE.MT.SCRY.SEQY ACE.MT.SCRX.AWX ACE.MT.SCRY.AWY ACE.MT.SEQ.SCR.REGION
ACE.MT.SEQ.AW.REGION ACE.MT.AW.SCR.POINT ACE.MT.AWX.SCRX ACE.MT.AWY.SCRY ACE.MT.AWX.SEQX
ACE.MT.AWY.SEQY ACE.MT.SEQX.SCRX ACE.MT.SEQY.SCRY ACE.MT.SEQX.AWX ACE.MT.SEQY.AWY
ACE.MAC.CW.INFO.CLIP ACE.MAC.CW.PROMPT.CLIP ACE.MAC.SEQ.CLIP ACE.MAC.FETCH.WIDTH
ACE.MAC.FETCH.HEIGHT)

previous date: 16-Nov-93 14:13:50 {DSK}<Users>kaplan>Local>medley3.5>my-medley>lispusers>ACE>ACE.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

::
:: Copyright (c) 1988, 1993 by Michel Denber.

(RPAQQ **ACECOMS**

(
;; Animation Compiler and Environment

;; THIS IS THE START UP FILE FOR THE ACE SYSTEM

(DECLARE%: DONTCOPY (RECORDS ACE.FRAME ACE.BLIT)
(MACROS ACE.MT.SCRX.SEQX ACE.MT.SCRY.SEQY ACE.MT.SCRX.AWX ACE.MT.SCRY.AWY ACE.MT.SEQ.SCR.REGION
ACE.MT.SEQ.AW.REGION ACE.MT.AW.SCR.POINT ACE.MT.AWX.SCRX ACE.MT.AWY.SCRY ACE.MT.AWX.SEQX
ACE.MT.AWY.SEQY ACE.MT.SEQX.SCRX ACE.MT.SEQY.SCRY ACE.MT.SEQX.AWX ACE.MT.SEQY.AWY))

;; ANIMATION FILES

(COMS * ACE-MAINCOMS)
(COMS * ACE-PRIMCOMS)
(COMS * ACE-EDITCOMS))

;; Animation Compiler and Environment

;; THIS IS THE START UP FILE FOR THE ACE SYSTEM

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD ACE.FRAME (DELAY BLITS))

(RECORD ACE.BLIT (BITMAP XCOORD . YCOORD))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS **ACE.MT.SCRX.SEQX MACRO** ((SCREENXCOORD)
(IDIFFERENCE [IDIFFERENCE SCREENXCOORD (DSPXOFFSET NIL (WINDOWPROP
ACE.SEQ.WINDOW
'DSP]
ACE.SEQ.WINDOW.XOFF)))

(PUTPROPS **ACE.MT.SCRY.SEQY MACRO** ((SCREENYCOORD)
(IDIFFERENCE [IDIFFERENCE SCREENYCOORD (DSPYOFFSET NIL (WINDOWPROP
ACE.SEQ.WINDOW
'DSP]
ACE.SEQ.WINDOW.YOFF)))

(PUTPROPS **ACE.MT.SCRX.AWX MACRO** [(SCREENXCOORD)
(IDIFFERENCE SCREENXCOORD (DSPXOFFSET NIL (WINDOWPROP ACE.SEQ.WINDOW
'DSP])

(PUTPROPS **ACE.MT.SCRY.AWY MACRO** [(SCREENYCOORD)
(IDIFFERENCE SCREENYCOORD (DSPYOFFSET NIL (WINDOWPROP ACE.SEQ.WINDOW
'DSP])

(PUTPROPS **ACE.MT.SEQ.SCR.REGION MACRO** (NIL (CREATEREGION (ACE.MT.SEQX.SCRX 0)
(ACE.MT.SEQY.SCRY 0)
ACE.SEQ.WIDTH ACE.SEQ.HEIGHT)))

(PUTPROPS **ACE.MT.SEQ.AW.REGION MACRO** (NIL (CREATEREGION ACE.SEQ.WINDOW.XOFF ACE.SEQ.WINDOW.YOFF ACE.SEQ.WIDTH
ACE.SEQ.HEIGHT)))

(PUTPROPS **ACE.MT.AW.SCR.POINT MACRO** [(POINT)
(CONS (ACE.MT.AWX.SCRX (CAR POINT))
(ACE.MT.AWY.SCRY (CDR POINT))

(PUTPROPS **ACE.MT.AWX.SCRX MACRO** [(WINDOWXCOORD)
(IPLUS WINDOWXCOORD (DSPXOFFSET NIL (WINDOWPROP ACE.SEQ.WINDOW 'DSP])

(PUTPROPS **ACE.MT.AWY.SCRY MACRO** [(WINDOWYCOORD)
(IPLUS WINDOWYCOORD (DSPYOFFSET NIL (WINDOWPROP ACE.SEQ.WINDOW 'DSP])

```

(PUTPROPS ACE.MT.AWX.SEQX MACRO ((WINDOWX)
                                   (IDIFFERENCE WINDOWX ACE.SEQ.WINDOW.XOFF)))

(PUTPROPS ACE.MT.AWY.SEQY MACRO ((WINDOWY)
                                   (IDIFFERENCE WINDOWY ACE.SEQ.WINDOW.YOFF)))

(PUTPROPS ACE.MT.SEQX.SCRX MACRO ((SEQXCOORD)
                                   (IPLUS ACE.SEQ.WINDOW.XOFF (ACE.MT.AWX.SCRX SEQXCOORD)))

(PUTPROPS ACE.MT.SEQY.SCRY MACRO ((SEQYCOORD)
                                   (IPLUS ACE.SEQ.WINDOW.YOFF (ACE.MT.AWY.SCRY SEQYCOORD)))

(PUTPROPS ACE.MT.SEQX.AWX MACRO ((SEQXCOORD)
                                   (IPLUS SEQXCOORD ACE.SEQ.WINDOW.XOFF)))

(PUTPROPS ACE.MT.SEQY.AWY MACRO ((SEQYCOORD)
                                   (IPLUS SEQYCOORD ACE.SEQ.WINDOW.YOFF)))
)
)

```

:: ANIMATION FILES

```

(RPAQQ ACE-MAINCOMS
  [(* MAIN TOP LEVEL STUFF)
   (FNS ACE ACE.ANIMATE ACE.RUN ACEGETFRAME# ACERUNLOOP ACE.NEW.SEQUENCE ACE.NEW.FRAME ACE.QUIT.ACE
     ACE.RESET.SEQ ACE.RUN.CURRENT.SEQ ACE.DELAY ACE.DELAY.FRAME ACE.DELAY.SEQ ACE.DECREMENT.FRAME
     ACE.INCREMENT.FRAME ACE.DELETE.FRAME ACE.SET.DEVICE ACE.QUICKDRAW&UPD ACE.RECONSTRUCT.FRAME SUBLIST
   )
   (* TRILLIUM STUFF)
   (FNS ACE.TRILLIUM ACE.TRILLIUM.LOOP ACE.RUN.TRILLIUM ACE.QUIT.TRILLIUM ACE.CREATE.EDITING.BORDER)
   (* I/O STUFF)
   (FNS ACE.GET.SEQ.FILE ACE.PUT.SEQ.FILE ACE.GET.A.FILE.NAME)
   (* HELPER FNS)
   (FNS ACE.ASKEM ACE.TELLEM ACE.CONFIRMIT ACE.DEFINE.SEQ.WINDOW ACE.FIGURE.OUT.WINDOW
     ACE.RETURN.CLOSEST.VERTEX ACE.NEW.SEQ.ASST ACE.DELAY.FRAME.ASST ACE.SETUP.CW.CLIPPING.REGIONS
     ACE.CHECKSTUFF ACE.UPD.CONTROL.WINDOW ACE.UPD.CW.MULE ACE.UPD.CLEAR.SET.LINE
     ACE.CREATE.CONTROL.MENU ACE.SEQ.FETCH.WIDTH ACE.SEQ.FETCH.HEIGHT ACE.SET.SEQ.CLIP.REGION ACE.ASKEM2
     ACE.TELLEM2 ACE.UPD.CONTROL.WINDOW2)
   (* The following Macros set up restricting clipping regions)
   (MACROS ACE.MAC.CW.INFO.CLIP ACE.MAC.CW.PROMPT.CLIP ACE.MAC.SEQ.CLIP)
   (MACROS ACE.MAC.FETCH.WIDTH ACE.MAC.FETCH.HEIGHT)
   (CURSORS ACE.LEFTMOUSE.CURSOR ACE.MIDDLEMOUSE.CURSOR ACE.RIGHTMOUSE.CURSOR ACE.ALLMOUSE.CURSOR)
   (GLOBALVARS ACE.CONTROL.WINDOW ACE.DIRECTORY ACE.SEQ.WINDOW ACE.SEQ.WIDTH ACE.SEQ.HEIGHT
     ACE.SEQ.WINDOW.XOFF ACE.SEQ.WINDOW.YOFF ACE.CURRENT.SEQUENCE ACE.CURRENT.SEQUENCE.NAME
     ACE.FRAME.TAIL ACE.CURRENT.FRAME ACE.VERTICAL.BLOCK ACE.AREA.THRESHOLD ACE.RUNNING.UNDER.TRILLIUM
     ACE.LEFTMOUSE.CURSOR ACE.MIDDLEMOUSE.CURSOR ACE.RIGHTMOUSE.CURSOR ACE.ALLMOUSE.CURSOR)
   (* MENUS IN MAIN)
   (GLOBALVARS ACE.CONTROL.MENU ACE.DELAY.MENU ACE.SET.DEVICE.MENU)
   (P (SETQ ACE.CONTROL.WINDOW NIL)
      (SETQ ACE.CONTROL.MENU NIL)
      (SETQ ACE.DELAY.MENU NIL)
      (SETQ ACE.SET.DEVICE.MENU NIL))
   (DECLARE%: DONTVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILEVARS (ADDVARS (NLAMA)
                                                                    (NLAML)
                                                                    (LAMA]))

```

(* MAIN TOP LEVEL STUFF)

(DEFINEQ

(ACE

[LAMBDA (SEQUENCE WINDOW POSITION APPLICATION)

(* MD "14-Jun-85 17:34")

(* Top level function to run animation. All ARGs are optional (No ARGs means just run "normal" ACE);
Current APPLICATIONs are NIL (Normal) and TRILLIUM. If TRILLIUM, then POSITION is necessary
(and WINDOW very highly recommended!); ACE creates ACE.CONTROL.WINDOW and menu if necessary;
then decides about APPLICATION)

```

(PROG (FONT TEMP.REGION)
  (OR ACE.CONTROL.MENU (ACE.CREATE.CONTROL.MENU))
  (SETQ FONT (FONTCREATE 'HELVETICA 10 'BOLD))
  (COND
    ((WINDOWP ACE.CONTROL.WINDOW)
     (TOTOPW ACE.CONTROL.WINDOW))
    (T (SETQ ACE.CONTROL.WINDOW (CREATEW (LIST 500 500 (IPLUS 2 (fetch IMAGEWIDTH of ACE.CONTROL.MENU))
                                             (IPLUS (ITIMES 4 (FONTPROP FONT 'HEIGHT))
                                             (fetch IMAGEHEIGHT of ACE.CONTROL.MENU)
                                             25))
      (CONCAT "ACE v. " "2.1" " Control Window")
      1))
    (DSPFONT FONT ACE.CONTROL.WINDOW)
    (WINDOWPROP ACE.CONTROL.WINDOW 'RESHAPEFN 'DON'T)
    (WINDOWPROP ACE.CONTROL.WINDOW 'REPAINTFN 'DON'T)
    (ACE.SETUP.CW.CLIPPING.REGIONS ACE.CONTROL.MENU)))
  (ACE.TELLEM "Position This Window" T)

```

```

(WINDOWPROP ACE.CONTROL.WINDOW 'MOVEFN NIL)
(MOVEW ACE.CONTROL.WINDOW)
(SETQ ACE.AREA.THRESHOLD 50)
(SETQ ACE.VERTICAL.BLOCK 16)
(SETQ ACE.CURRENT.SEQUENCE.NAME NIL)
(COND
  ((EQ APPLICATION 'TRILLIUM)
   (RETURN (ACE.TRILLIUM WINDOW SEQUENCE POSITION)))
  (T (SETQ ACE.RUNNING.UNDER.TRILLIUM NIL)
     (SETQ ACE.DIRECTORY (ACE.ASKEM "Animation Directory? " T LOGINHOST/DIR))
     (ACE.TELLEM "If using the Tablet," T)
     (ACE.TELLEM "Be sure and Initialize it." 'L)
     (SETQ ACE.SEQ.WINDOW NIL)
     (SETQ ACE.SEQ.WIDTH NIL)
     (SETQ ACE.SEQ.HEIGHT NIL)
     (SETQ ACE.SEQ.WINDOW.XOFF 0)
     (SETQ ACE.SEQ.WINDOW.YOFF 0)
     (SETQ ACE.CURRENT.SEQUENCE NIL)
     (SETQ ACE.FRAME.TAIL NIL)
     (SETQ ACE.CURRENT.FRAME NIL)
     (replace (MENU WHENSELECTEDFN) of ACE.CONTROL.MENU with 'ACE.ANIMATE)
     (ADDMENU ACE.CONTROL.MENU ACE.CONTROL.WINDOW (CONS 0 0))
     (ACE.UPD.CONTROL.WINDOW 'RESET)))

```

(ACE.ANIMATE

[LAMBDA (ITEM WHO CARES)

(* MD "26-Jun-85 14:21")

(* When running "normal" ACE, this is the WHENSELECTEDFN for deciding what to do.
 ITEM is the only ARG of significance (WHO & CARES just to keep MENU package happy);
 "TRILLIUM ACE" has its own loop thingy)

```

(ACE.TELLEM "" T)
(SELECTQ (CADADR ITEM)
  (NIL NIL)
  (FRAME (ACE.NEW.FRAME))
  (EDIT (ACE.EDIT))
  (RUN (ACE.RUN.CURRENT.SEQ))
  (LOOP (ACERUNLOOP 1 'END))
  (SUBLOOP (ACEGETFRAME#))
  (NEW (ACE.NEW.SEQUENCE))
  (I/O%:GET (ACE.GET.SEQ.FILE))
  (I/O%:PUT (ACE.PUT.SEQ.FILE))
  (DEL (ACE.DELETE.FRAME))
  (RESET (ACE.RESET.SEQ))
  (INC (ACE.INCREMENT.FRAME))
  (DEC (ACE.DECREMENT.FRAME))
  (TIME (ACE.DELAY))
  (INIT.MM1201 (ACE.MM1201.INIT T))
  (DEVICE (ACE.SET.DEVICE))
  (ACESETTHRESHOLD
   (ACESETTHRESHOLD))
  (QUIT (ACE.QUIT.ACE))
  NIL)
(AND (OPENWP ACE.CONTROL.WINDOW)
     (ACE.UPD.CONTROL.WINDOW 'UPD))
(TTY.PROCESS T))

```

(ACE.RUN

[LAMBDA (SEQ WINDOW XOFFSET YOFFSET TIMER)

(* PmT "18-Apr-85 18:21")

(* This runs animation sequences. Simple, eh? Gots to have a SEQ and a WINDOW to show it in, and where in the window to show it (i.e. XOFFSET YOFFSET); TIMER is wholly unnecessary NOTE%: this thang doesn't use any clipping region, just an offset)

```

(for FRAME in SEQ do ((SETQ TIMER (SETUPTIMER (fetch (ACE.FRAME DELAY) of FRAME)
                                              TIMER
                                              'MILLISECONDS))
  (for FRAME.PART in (fetch (ACE.FRAME BLITS) of FRAME)
   do (BITBLT (fetch (ACE.BLIT BITMAP) of FRAME.PART)
              NIL NIL WINDOW (IPLUS XOFFSET (fetch (ACE.BLIT XCOOR) of FRAME.PART))
              (IPLUS YOFFSET (fetch (ACE.BLIT YCOOR) of FRAME.PART))
              NIL NIL 'INPUT 'REPLACE))
   (until (TIMEREXPIRED? TIMER 'MILLISECONDS) do))

```

(ACEGETFRAME#

[LAMBDA NIL

(* MD "27-Jun-85 12:51")

```

(ACERUNLOOP (RNUMBER "From frame #")
  (RNUMBER "To frame #:"))

```

(ACERUNLOOP

[LAMBDA (START END)

(* MD "26-Jun-85 17:07")

```

(if (EQ END 'END)
  then (SETQ END (LENGTH ACE.CURRENT.SEQUENCE)))

```

```
[ACE.QUICKDRAW&UPD (LIST (CAR (NTH ACE.CURRENT.SEQUENCE START)
(until (KEYDOWNP 'SPACE) do (ACE.RUN (SUBLIST ACE.CURRENT.SEQUENCE START END)
ACE.SEQ.WINDOW ACE.SEQ.WINDOW.XOFF ACE.SEQ.WINDOW.YOFF])
```

(ACE.NEW.SEQUENCE

(* PmT "30-Apr-85 16:21")

```
[LAMBDA (REGION)
  (PROG (TEMP.REGION)
    (ACE.UPD.CONTROL.WINDOW 'OPERATION "NEW")
    (COND
      ((OR (NULL ACE.CURRENT.SEQUENCE)
        (ACE.TELLEM "Creating a NEW Sequence will ERASE" T)
        (ACE.TELLEM "the Current Animation Sequence." 'L)
        (ACE.CONFIRMIT "Click LEFT to Create NEW Sequence." 'L))
      [COND
        ((REGIONP REGION)
          (SETQ TEMP.REGION REGION))
        (T (ACE.TELLEM "Specify SIZE of the NEW Sequence." T)
          (ACE.TELLEM "Watch ACE window for COORs." 'L)
          (SETQ TEMP.REGION (GETREGION NIL NIL NIL 'ACE.NEW.SEQ.ASST NIL)
            (SETQ ACE.SEQ.WIDTH (fetch (REGION WIDTH) of TEMP.REGION))
            (SETQ ACE.SEQ.HEIGHT (fetch (REGION HEIGHT) of TEMP.REGION))
            (SETQ ACE.CURRENT.SEQUENCE (LIST (create ACE.FRAME
              DELAY _ 0
              BLITS _ (LIST (create ACE.BLIT
                BITMAP _ (BITMAPCREATE ACE.SEQ.WIDTH
                  ACE.SEQ.HEIGHT 1)
                XCOORD _ 0
                YCOORD _ 0)
              (ACE.FIGURE.OUT.WINDOW)
              (ACE.QUICKDRAW&UPD (LIST (CAR ACE.CURRENT.SEQUENCE]))
```

(ACE.NEW.FRAME

(* PmT "24-Jan-85 13:22")

```
[LAMBDA NIL
  (ACE.UPD.CONTROL.WINDOW 'OPERATION "FRAME")
  (COND
    ([NULL (AND (ACE.CHECKSTUFF 'SEQ)
      (OR (ACE.CHECKSTUFF 'FRAME)
        (ACE.TELLEM "Can't put a frame before the First Frame." T)
      NIL)
    (T (RPLACD ACE.CURRENT.FRAME (CONS (create ACE.FRAME
      DELAY _ 0
      BLITS _ NIL)
      (CDR ACE.CURRENT.FRAME)))
      (SETQ ACE.CURRENT.FRAME (CDR ACE.CURRENT.FRAME))
      (SETQ ACE.FRAME.TAIL (CDR ACE.CURRENT.FRAME))
      (ACE.TELLEM "Going to EDIT ..." T)
      (ACE.EDIT])
```

(ACE.QUIT.ACE

(* MD "24-Jun-85 14:49")

```
[LAMBDA NIL
  (ACE.UPD.CONTROL.WINDOW 'OPERATION "QUIT")
  (ACE.TELLEM "QUITting will Close All Animation Windows;" T)
  (ACE.TELLEM "All Images and Data will be LOST." 'L)
  (COND
    ((ACE.CONFIRMIT "Click LEFT to QUIT." 'L)
      (CLOSEW ACE.SEQ.WINDOW)
      (CLOSEW ACE.CONTROL.WINDOW)
      ACE.CURRENT.SEQUENCE)
    (T
      (* MAKE THIS BE SOME KINDA ICON IN THE FUTURE; ASK%: DO YOU WANT TO QUIT COMPLETELY OR JUST
      STOP FOR A WHILE)
      (ACE.TELLEM "QUIT Aborted." T])
```

(ACE.RESET.SEQ

(* PmT "30-Apr-85 16:37")

```
[LAMBDA NIL
  (ACE.UPD.CONTROL.WINDOW 'OPERATION "RESET")
  (COND
    [(AND ACE.SEQ.WINDOW (ACE.CHECKSTUFF 'SEQ))
      (ACE.QUICKDRAW&UPD (LIST (CAR ACE.CURRENT.SEQUENCE)
      (T (ACE.TELLEM "There is No Current Sequence." T])
```

(ACE.RUN.CURRENT.SEQ

(* PmT "18-Apr-85 18:23")

(* just a pretty interface to ACE.RUN)

```
[LAMBDA NIL
  (ACE.UPD.CONTROL.WINDOW 'OPERATION "RUN")
  (COND
    ((ACE.CHECKSTUFF 'SEQ)
      (RECLAIM)
      (ACE.RUN ACE.FRAME.TAIL ACE.SEQ.WINDOW ACE.SEQ.WINDOW.XOFF ACE.SEQ.WINDOW.YOFF)
      (SETQ ACE.FRAME.TAIL NIL)
      (SETQ ACE.CURRENT.FRAME (LAST ACE.CURRENT.SEQUENCE]))
```

(ACE.DELAY

[LAMBDA NIL

(* PmT "2-May-85 20:53")

(* For setting delays (in MSECs) between frames.

Lots of work needed here; esp. delay in-betweening)

(ACE.UPD.CONTROL.WINDOW 'OPERATION "DELAY")

(SELECTQ [MENU (OR ACE.DELAY.MENU (SETQ ACE.DELAY.MENU (create MENU

```

ITEMS _ ' (("Set Delay on a Frame" 'FDELAY
            "Adjust the delay on any
            frame by number")
            ("Reset Entire Sequence"
            'SDELAY "Set the delay on every
            frame"))
CENTERFLG _ T
TITLE _ "Sequence Timing Adjustments"]

```

(NIL NIL)

(SDELAY (ACE.DELAY.SEQ))

(FDELAY (ACE.DELAY.FRAME))

NIL])

(ACE.DELAY.FRAME

[LAMBDA NIL

(* MD "21-Jun-85 14:14")

(PROG (CHOICE FRAME)

LOOP

[SETQ CHOICE (MENU (create MENU

ITEMS _ (NCONC1 (for FRAME in ACE.CURRENT.SEQUENCE

bind (COUNT _ 0)

collect ((SETQ COUNT (ADD1 COUNT))

(LIST (CONCAT "Frame " COUNT " : "

(fetch (ACE.FRAME DELAY) of FRAME))

COUNT)))

' (Quit 'QUIT "Stop adjusting delays"))

TITLE _ "Frame Delays"

WHENHELDFN _ 'ACE.DELAY.FRAME.ASST]

[COND

((NULL CHOICE)

NIL)

((EQ CHOICE 'QUIT)

(RETURN NIL))

(T (SETQ FRAME (CAR (NTH ACE.CURRENT.SEQUENCE CHOICE)))

(AND FRAME (replace (ACE.FRAME DELAY) of FRAME with (SETQ CHOICE (RNUMBER (CONCAT "Frame " CHOICE

"; New Delay: ")

(GO LOOP]))

(ACE.DELAY.SEQ

[LAMBDA NIL

(* MD "21-Jun-85 14:31")

(PROG (NEW.DELAY.VALUE)

(COND

[(FIXP (SETQ NEW.DELAY.VALUE (RNUMBER "Delay for entire sequence:"))

(T (RETURN NIL)))

(for FRAME in ACE.CURRENT.SEQUENCE do (replace (ACE.FRAME DELAY) of FRAME with NEW.DELAY.VALUE))

(ACE.DECREMENT.FRAME

[LAMBDA NIL

(* PmT "21-Dec-84 14:12")

(ACE.UPD.CONTROL.WINDOW 'OPERATION "DEC")

(AND (ACE.CHECKSTUFF 'SEQ)

(ACE.CHECKSTUFF 'FRAME)

(ACE.QUICKDRAW&UPD (LDIFF ACE.CURRENT.SEQUENCE ACE.CURRENT.FRAME]))

(ACE.INCREMENT.FRAME

[LAMBDA NIL

(* MD "18-Jun-85 16:12")

(PROG (CUR.FRAME)

(ACE.UPD.CONTROL.WINDOW 'OPERATION "INC")

(COND

[(NULL (AND (ACE.CHECKSTUFF 'SEQ)

(ACE.CHECKSTUFF 'TAIL])

NIL)

(T (SETQ CUR.FRAME (CAR ACE.FRAME.TAIL))

(SETQ ACE.CURRENT.FRAME ACE.FRAME.TAIL)

(SETQ ACE.FRAME.TAIL (CDR ACE.FRAME.TAIL))

(COND

((NULL (fetch (ACE.FRAME BLITS) of CUR.FRAME))

NIL)

(T (ACE.MAC.SEQ.CLIP (for FRAME.PART in (fetch (ACE.FRAME BLITS) of CUR.FRAME)

do (BITBLT (fetch (ACE.BLIT BITMAP) of FRAME.PART)

NIL NIL ACE.SEQ.WINDOW (IPLUS ACE.SEQ.WINDOW.XOFF

(fetch (ACE.BLIT XCOORD)

of FRAME.PART))

(IPLUS ACE.SEQ.WINDOW.YOFF (fetch (ACE.BLIT YCOORD)

of FRAME.PART))

NIL NIL 'INPUT 'REPLACE]))

(ACE.DELETE.FRAME

[LAMBDA NIL

(* PmT "24-Apr-85 14:19")

(* Deletes the current frame; recompiles the previous frame with

the successor frame)

(PROG (BEFORE.BM AFTER.BM)

(ACE.UPD.CONTROL.WINDOW 'OPERATION "DELETE")

(COND

([NULL (AND (ACE.CHECKSTUFF 'SEQ)

(ACE.CHECKSTUFF 'FRAME]

NIL)

(EQ ACE.CURRENT.SEQUENCE ACE.CURRENT.FRAME)

(ACE.TELLEM "Can't DELETE first frame. Aborted." T))

(NULL (ACE.CONFIRMIT "Click LEFT to Confirm Delete" T))

NIL)

(NULL ACE.FRAME.TAIL)

(SETQ ACE.CURRENT.SEQUENCE (LDIFF ACE.CURRENT.SEQUENCE ACE.CURRENT.FRAME)))

(ACE.QUICKDRAW&UPD ACE.CURRENT.SEQUENCE))

(T (SETQ BEFORE.BM (ACE.RECONSTRUCT.FRAME (LDIFF ACE.CURRENT.SEQUENCE ACE.CURRENT.FRAME)))

[SETQ AFTER.BM (ACE.RECONSTRUCT.FRAME (LDIFF ACE.CURRENT.SEQUENCE (CDR ACE.FRAME.TAIL]

(replace (ACE.FRAME BLITS) of (CAR ACE.CURRENT.FRAME) with (ACE.COMPILE.FRAME BEFORE.BM AFTER.BM

ACE.VERTICAL.BLOCK

ACE.AREA.THRESHOLD))

(RPLACD ACE.CURRENT.FRAME (CDR ACE.FRAME.TAIL))

(ACE.QUICKDRAW&UPD (LDIFF ACE.CURRENT.SEQUENCE (CDR ACE.CURRENT.FRAME]))

(ACE.SET.DEVICE

[LAMBDA NIL

(* PmT "23-Apr-85 13:44")

(* Selects MOUSE or TABLET as the primary input device)

(ACE.UPD.CONTROL.WINDOW 'OPERATION "SET DEVICE")

(ACE.UPD.CONTROL.WINDOW 'DEVICE)

(ACE.UPD.CONTROL.WINDOW 'DEVICE (MENU (OR ACE.SET.DEVICE.MENU (SETQ ACE.SET.DEVICE.MENU

(create MENU

ITEMS _ ' ("Mouse" 'MOUSE "Use the
standard mouse for
drawing and such")

("Tablet" 'MM1201 "Use the

MM1201 Tablet as

the input device"))

TITLE _ "Select Input Device"

CENTERFLG _ T])

(ACE.QUICKDRAW&UPD

[LAMBDA (PARTIAL.SEQ)

(* PmT "30-Apr-85 16:11")

(* Updates the frame showing in the A.S.Window and update sequence pointers and stuff.

PARTIAL.SEQ is a list of frames to show; The last frame in PARTIAL.SEQ becomes the new current frame)

(COND

(PARTIAL.SEQ [ACE.MAC.SEQ.CLIP (for FRAME in PARTIAL.SEQ

do (COND

((NULL (fetch (ACE.FRAME BLITS) of FRAME))

NIL)

(T (for FRAME.PART in (fetch (ACE.FRAME BLITS) of FRAME)

do (BITBLT (fetch (ACE.BLIT BITMAP) of FRAME.PART)

NIL NIL ACE.SEQ.WINDOW (IPLUS ACE.SEQ.WINDOW.XOFF

(fetch (ACE.BLIT

XCOORD)

of FRAME.PART))

(IPLUS ACE.SEQ.WINDOW.YOFF (fetch (ACE.BLIT YCOORD)

of FRAME.PART))

NIL NIL 'INPUT 'REPLACE]

(SETQ ACE.CURRENT.FRAME ACE.CURRENT.SEQUENCE)

(for X from 1 to (SUB1 (LENGTH PARTIAL.SEQ)) do (SETQ ACE.CURRENT.FRAME (CDR ACE.CURRENT.FRAME)))

(SETQ ACE.FRAME.TAIL (CDR ACE.CURRENT.FRAME]))

(ACE.RECONSTRUCT.FRAME

[LAMBDA (SEQ)

(* PmT "18-Apr-85 18:54")

(* Creates a bitmap out of SEQ; Essentially, the last virtual frame in SEQ is converted to a "real" frame and returned)

(PROG (ABITMAP)

[SETQ ABITMAP (BITMAPCOPY (fetch (ACE.BLIT BITMAP) of (CAR (fetch (ACE.FRAME BLITS) of (CAR SEQ)

[for FRAME in (CDR SEQ) do (COND

(NULL (fetch (ACE.FRAME BLITS) of FRAME)))

(T (for FRAME.PART in (fetch (ACE.FRAME BLITS) of FRAME)

do (BITBLT (fetch (ACE.BLIT BITMAP) of FRAME.PART)

NIL NIL ABITMAP (fetch (ACE.BLIT XCOORD) of FRAME.PART)

(fetch (ACE.BLIT YCOORD) of FRAME.PART)

NIL NIL 'INPUT 'REPLACE]

(RETURN ABITMAP))

(SUBLIST

```

[LAMBDA (L M N)
  (LDIFF (NTH L M)
    (NTH L (ADD1 N)))
)
(* MD "26-Jun-85 16:00")

```

```

(* * TRILLIUM STUFF)

```

```

(DEFINEQ

```

(ACE.TRILLIUM

```

[LAMBDA (WINDOW SEQUENCE POSITION)
(* PmT "30-Apr-85 16:44")

```

(* This here sets up stuff for running animation (functionally) from Trillium.
 ARGS%: WINDOW is opt (but really should be given)%, SEQUENCE is opt, POSITION *MUST* be given;
 if not, ACE bags it (Trillium must always supply a place to put the animation!)%: This FN just checks args and sets the state
 of ACE, then calls the actual "polling" FN NOTE%: This should only be called from ACE;
 take a look at ACE)

```

(PROG NIL
  (SETQ ACE.RUNNING.UNDER.TRILLIUM T)
  (DISPLAY.FRAME CURRENT.FRAME)
  [COND
    ((WINDOWP WINDOW)
      (SETQ ACE.SEQ.WINDOW WINDOW))
    (T (ACE.TELLEM "There is no Window Specification" T)
      (COND
        ((ACE.CONFIRMIT "Click LEFT to use Current Interface Window." 'L)
          (SETQ ACE.SEQ.WINDOW CURRENT.INTERFACE.WINDOW))
        (T (ACE.TELLEM "Error in Window Specification." T)
          (ACE.TELLEM "Likely a Trillium error. ACE aborted." 'L)
          (ACE.CONFIRMIT "Click any button to Exit." 'L 'ANY)
          (RETURN NIL)
          (* The following *might* be used instead of a window error;
            you decide)
          (* SETQ ACE.SEQ.WINDOW (EVAL
            (ACE.ASKEM "Enter the Name of the Window: " T NIL 60)))
          (* Was given a valid postion?))
        ))
    (COND
      ((POSITIONP POSITION)
        (SETQ ACE.SEQ.WINDOW.XOFF (fetch (POSITION XCOORD) of POSITION))
        (SETQ ACE.SEQ.WINDOW.YOFF (fetch (POSITION YCOORD) of POSITION)))
      (T (ACE.TELLEM "No Position Specification. Aborted." T)
        (ACE.TELLEM "This is likely a Trillium error." 'L)
        (ACE.CONFIRMIT "Click any button to Exit." 'L 'ANY)
        (RETURN NIL)))
    (COND
      ((LISTP SEQUENCE)
        (SETQ ACE.CURRENT.SEQUENCE SEQUENCE)
        (SETQ ACE.CURRENT.FRAME ACE.CURRENT.SEQUENCE)
        (SETQ ACE.FRAME.TAIL (CDR ACE.CURRENT.FRAME))
        (SETQ ACE.SEQ.WIDTH (ACE.SEQ.FETCH.WIDTH))
        (SETQ ACE.SEQ.HEIGHT (ACE.SEQ.FETCH.HEIGHT))
        (ACE.CREATE.EDITING.BORDER)
        (ACE.SET.SEQ.CLIP.REGION))
      (T (SETQ ACE.CURRENT.SEQUENCE NIL)
        (SETQ ACE.FRAME.TAIL NIL)
        (SETQ ACE.CURRENT.FRAME NIL)))
    (replace (MENU WHENSELECTEDFN) of ACE.CONTROL.MENU with 'DEFAULTWHENSELECTEDFN)
    (SETQ ACE.DIRECTORY (DIRECTORYNAME T T))
    (AND ACE.CURRENT.SEQUENCE (ACE.RESET.SEQ))
    (ACE.UPD.CONTROL.WINDOW 'RESET)
    (WINDOWPROP ACE.CONTROL.WINDOW 'MOVEFN 'DON'T)
    (RETURN (ACE.TRILLIUM.LOOP))
    (* Lock down window so menu coors only figured once;
    see ACE.TRILLIUM.LOOP)
  )

```

(ACE.TRILLIUM.LOOP

```

[LAMBDA NIL
(* PmT "18-Apr-85 18:41")
(* This is the repeating loop for Trillium-Ace;
  just sits in here till QUIT)

```

```

(PROG (CHOICE MENU.POS)
  [SETQ MENU.POS (CONS (DSPXOFFSET NIL (WINDOWPROP ACE.CONTROL.WINDOW 'DSP))
    (DSPYOFFSET NIL (WINDOWPROP ACE.CONTROL.WINDOW 'DSP)])
  LOOP
    (ACE.TELLEM "" T)
    (SELECTQ (SETQ CHOICE (MENU ACE.CONTROL.MENU MENU.POS))
      (NIL NIL)
      (FRAME (ACE.NEW.FRAME))
      (EDIT (ACE.EDIT))
      (RUN (ACE.RUN.CURRENT.SEQ))
      (NEW (ACE.NEW.SEQUENCE))
      (I/O%:GET (ACE.GET.SEQ.FILE))
      (I/O%:PUT (ACE.PUT.SEQ.FILE))
      (DEL (ACE.DELETE.FRAME))
      (RESET (ACE.RESET.SEQ))
    )
  )

```

```

(inc (ACE.INCREMENT.FRAME))
(dec (ACE.DECREMENT.FRAME))
(time (ACE.DELAY))
(init.MM1201 (ACE.MM1201.INIT T))
(device (ACE.SET.DEVICE))
(quit nil)
nil)
(ACE.UPD.CONTROL.WINDOW 'UPD)
(or (eq choice 'QUIT)
    (go loop))
(return (ACE.QUIT.TRILLIUM])

```

(ACE.RUN.TRILLIUM

[LAMBDA (SEQ WINDOW XOFFSET YOFFSET UPTO TIMER)

(* PmT "18-Apr-85 18:45")

(* Just like ACE.RUN except UPTO can be a FIXP denoting a frame;
 If UPTO is given, that frame is displayed (without delays); Good for initializing in Trillium)

```

(COND
  [(NULL UPTO)
   (for FRAME in SEQ do ((SETQ TIMER (SETUPTIMER (fetch (ACE.FRAME DELAY) of FRAME)
                                                    TIMER
                                                    'MILLISECONDS))
                        (for FRAME.PART in (fetch (ACE.FRAME BLITS) of FRAME)
                          do (BITBLT (fetch (ACE.BLIT BITMAP) of FRAME.PART)
                                      NIL NIL WINDOW (IPLUS XOFFSET (fetch (ACE.BLIT XCOORD) of FRAME.PART))
                                      (IPLUS YOFFSET (fetch (ACE.BLIT YCOORD) of FRAME.PART))
                                      NIL NIL 'INPUT 'REPLACE))
                        (until (TIMEREXPIRED? TIMER 'MILLISECONDS) do]
   ((AND (FIXP UPTO)
        (IGREATERP (ADD1 (LENGTH SEQ))
                    UPTO)
        (IGREATERP UPTO 0))
    (for FRAME in (LDIFF SEQ (NTH SEQ (ADD1 UPTO)))
      do (for FRAME.PART in (fetch (ACE.FRAME BLITS) of FRAME) do (BITBLT (fetch (ACE.BLIT BITMAP) of
                                                                                     FRAME.PART)
                                                                                     NIL NIL WINDOW
                                                                                     (IPLUS XOFFSET (fetch (ACE.BLIT XCOORD)
                                                                                               of FRAME.PART))
                                                                                     (IPLUS YOFFSET (fetch (ACE.BLIT YCOORD)
                                                                                               of FRAME.PART))
                                                                                     NIL NIL 'INPUT 'REPLACE]))

```

(ACE.QUIT.TRILLIUM

```

[LAMBDA NIL
  (CLOSEW ACE.CONTROL.WINDOW)
  (SETQ ACE.RUNNING.UNDER.TRILLIUM NIL)
  ACE.CURRENT.SEQUENCE])

```

(* PmT "15-Mar-85 13:48")

(ACE.CREATE.EDITING.BORDER

```

[LAMBDA (MODE)
  (PROG (X1 X2 Y1 Y2)
    (OR MODE (SETQ MODE 'PAINT))
    (COND
      ((AND (NUMBERP ACE.SEQ.WIDTH)
            (NUMBERP ACE.SEQ.HEIGHT))
       (SETQ X1 (IDIFFERENCE ACE.SEQ.WINDOW.XOFF 2))
       (SETQ X2 (IPLUS ACE.SEQ.WINDOW.XOFF ACE.SEQ.WIDTH))
       (SETQ Y1 (IDIFFERENCE ACE.SEQ.WINDOW.YOFF 2))
       (SETQ Y2 (IPLUS ACE.SEQ.WINDOW.YOFF ACE.SEQ.HEIGHT))
       (DRAWLINE X1 Y1 X1 Y2 2 MODE ACE.SEQ.WINDOW)
       (DRAWLINE X1 Y2 X2 Y2 2 MODE ACE.SEQ.WINDOW)
       (DRAWLINE X2 Y2 X2 Y1 2 MODE ACE.SEQ.WINDOW)
       (DRAWLINE X2 Y1 X1 Y1 2 MODE ACE.SEQ.WINDOW])

```

(* PmT "30-Apr-85 16:42")

(* I/O STUFF)

(DEFINEQ

(ACE.GET.SEQ.FILE

[LAMBDA NIL

(* PmT "25-Apr-85 21:18")

(* Gets an animation sequence. Resets
 ACE.CURRENT.SEQUENCE and the sequence clipping region)

```

(RESETFORM (TTYDISPLAYSTREAM \TopLevelTtyWindow)
  (PROG (FILENAME TEMP.SEQUENCE.NAME)
    (ACE.UPD.CONTROL.WINDOW 'OPERATION "GET FILE")
    (OR (NULL ACE.CURRENT.SEQUENCE)
        (ACE.TELLEM "Loading a Sequence will ERASE the Current" T)
        (ACE.CONFIRMIT "Sequence; Click LEFT to confirm LOAD." 'L)
        (ACE.TELLEM "Get Sequence ABORTED." T)

```



```

(RETURN NIL))
(SETQ FILENAME (ACE.GET.A.FILE.NAME))
[COND
  ((NULL FILENAME)
   (ACE.TELLEM "No NAME. Aborted" T)
   (RETURN NIL))
  (T (SETQ FILENAME (PACKFILENAME 'BODY FILENAME 'HOST (FILENAMEFIELD ACE.DIRECTORY
                                                                    'HOST)
                                                                    'DIRECTORY
                                                                    (FILENAMEFIELD ACE.DIRECTORY 'DIRECTORY))
   (ACE.TELLEM "Loading: " T)
   (ACE.TELLEM (CONCAT FILENAME " ... ")
                'L)
   (RESETLST
    [RESETSAVE (PROGN (CURSOR WAITINGCURSOR)
                      (SETTOPVAL 'HELPFLAG NIL))
      (LIST 'PROGN (LIST 'CURSOR 'DEFAULTCURSOR)
            (LIST 'SETTOPVAL 'HELPFLAG (KWOTE (GETTOPVAL 'HELPFLAG)
            [SETQ TEMP.SEQUENCE.NAME (CAR (ERRORSET ' (LOAD FILENAME 'SYSLOAD)
            'NOBREAK]))
    (COND
      (TEMP.SEQUENCE.NAME (SETQ ACE.CURRENT.SEQUENCE.NAME TEMP.SEQUENCE.NAME)
      (SETQ ACE.SEQ.WIDTH (ACE.SEQ.FETCH.WIDTH))
      (SETQ ACE.SEQ.HEIGHT (ACE.SEQ.FETCH.HEIGHT))
      (ACE.FIGURE.OUT.WINDOW)
      (ACE.RESET.SEQ))
    (T (ACE.TELLEM "Not Found.")
      (ACE.TELLEM "No Such File or File Server Problems." 'L]))

```

(ACE.PUT.SEQ.FILE

[LAMBDA NIL

(* PmT "2-May-85 20:50")

(* Writes a sequence to a file; the file is NOT pretty printed)

```

(PROG (FILENAME TEMP.SEQUENCE.NAME)
  (ACE.UPD.CONTROL.WINDOW 'OPERATION "PUT FILE")
  [COND
    ((NULL (ACE.CHECKSTUFF 'SEQ))
     (RETURN NIL))
    (AND ACE.CURRENT.SEQUENCE.NAME (ACE.CONFIRMIT "Click LEFT to Keep Same Name." T))
    (SETQ FILENAME ACE.CURRENT.SEQUENCE.NAME))
  (T (SETQ FILENAME (ACE.GET.A.FILE.NAME))
    [COND
      ((NULL FILENAME)
       (ACE.TELLEM "NIL ain't no good. Aborted." T)
       (RETURN NIL))
      (SETQ FILENAME (PACKFILENAME 'BODY FILENAME 'HOST (FILENAMEFIELD ACE.DIRECTORY 'HOST)
                                'DIRECTORY
                                (FILENAMEFIELD ACE.DIRECTORY 'DIRECTORY))
    (COND
      ((AND (FILENAMEFIELD FILENAME 'VERSION)
        (NULL (ACE.TELLEM "Click LEFT to Write a New Version." T))
        (ACE.CONFIRMIT "Click any Other to Write Over Existing Version." 'L))
       (SETQ FILENAME (PACKFILENAME 'VERSION NIL 'BODY FILENAME)))
      (T NIL))
    [SET (PACK* (FILENAMEFIELD FILENAME 'NAME)
               'COMS)
        ' ((UGLYVARS ACE.CURRENT.SEQUENCE)
        (PUTPROP (FILENAMEFIELD FILENAME 'NAME)
                  'FILETYPE
                  ' (DON'TLIST DON'TCOMPILE))
        (ACE.TELLEM "Putting to File: " T)
        (ACE.TELLEM (CONCAT FILENAME " ... ")
                      'L)
        (RESETLST
         [RESETSAVE (PROGN (CURSOR WAITINGCURSOR)
                           (SETTOPVAL 'HELPFLAG NIL))
           (LIST 'PROGN (LIST 'CURSOR 'DEFAULTCURSOR)
                 (LIST 'SETTOPVAL 'HELPFLAG (KWOTE (GETTOPVAL 'HELPFLAG)
           [SETQ TEMP.SEQUENCE.NAME (CAR (ERRORSET ' (MAKEFILE FILENAME ' (NEW FAST))
           'NOBREAK]))
         (COND
           (TEMP.SEQUENCE.NAME (SETQ ACE.CURRENT.SEQUENCE.NAME TEMP.SEQUENCE.NAME)
           (ACE.TELLEM "Done")
           (DREMOVE (FILENAMEFIELD FILENAME 'NAME)
                    FILELST))
           (T (ACE.TELLEM "Aborted.")
             (ACE.TELLEM "Nothing doing. Can't write this File out." 'L)
             (ACE.TELLEM "Check the Name. Is the File Server Down?" 'L]))

```

(ACE.GET.A.FILE.NAME

[LAMBDA NIL

(* PmT "18-Apr-85 19:44")

(ACE.ASKEM "Enter FILENAME: " T NIL 120))

)

(* * HELPER FNS)

(DEFINEQ

(ACE.ASKEM

[LAMBDA (STRING FLG DEFAULTANSWER TIMELIMIT? SPACES?) (* MD "14-Jun-85 16:48")

(* a prompting fn. STRING is the prompt string; FLG either T,L or NIL
 (T means clear before prompting, L means new line); DEFAULTANSWER just what it sounds like.
 TIMELIMIT? number or seconds to wait for answer (defaults to 120);
 if SPACES? is T, then the answer can have spaces in it)

(* TIMELIMIT? removed -
 now waits forever. -
 MJD)

```
(ACE.MAC.CW.PROMPT.CLIP (PROGN (OR TIMELIMIT? (SETQ TIMELIMIT? 120))
                                (COND
                                 ((EQ FLG T)
                                  (DSPRESET ACE.CONTROL.WINDOW))
                                 ((EQ FLG 'L)
                                  (TERPRI ACE.CONTROL.WINDOW))))
  (MKATOM (PROMPTFORWARD STRING DEFAULTANSWER NIL ACE.CONTROL.WINDOW NIL NIL
                        (AND SPACES? (CHARCODE (EOL ESCAPE LF))
```

(ACE.TELLEM

[LAMBDA (STRING FLG) (* PmT "23-Apr-85 13:49")

(* Writes STRING in the A.C.W prompt region; FLG=T means clear prompt region first;
 L means new line; NIL means put it at the next char position)

```
(ACE.MAC.CW.PROMPT.CLIP (PROGN (COND
                                ((EQ FLG T)
                                 (DSPRESET ACE.CONTROL.WINDOW))
                                ((EQ FLG 'L)
                                 (TERPRI ACE.CONTROL.WINDOW))))
  (printout ACE.CONTROL.WINDOW STRING)
  NIL))
```

(ACE.CONFIRMIT

[LAMBDA (CONFIRMSTRING FLG WHICHKEYS?) (* PmT "25-Apr-85 17:47")

(* Prints CONFIRMSTRING in A.C.W prompt region; then waits for the button form WHICHKEYS? to become true.
 WHICHKEYS? defaults to LEFT. Code identifies the valid button forms)

```
(OR WHICHKEYS? (SETQ WHICHKEYS? 'LEFT))
(ACE.TELLEM CONFIRMSTRING FLG)
(DISSMISS 100 NIL T)
(RESETFORM (CURSOR (SELECTQ WHICHKEYS?
                             (LEFT ACE.LEFTMOUSE.CURSOR)
                             (MIDDLE ACE.MIDDLEMOUSE.CURSOR)
                             (RIGHT ACE.RIGHTMOUSE.CURSOR)
                             (ANY (PROGN (SETQ WHICHKEYS? ' (NOT UP))
                                           ACE.ALLMOUSE.CURSOR))
                             NIL))
  (do (GETMOUSESTATE) until (NEQ LASTMOUSEBUTTONS 0))
  (PROG1 (EVAL (MOUSESTATE-EXPR WHICHKEYS? T))
    (do (GETMOUSESTATE) until (EQP LASTMOUSEBUTTONS 0))))))
```

(ACE.DEFINE.SEQ.WINDOW

[LAMBDA NIL (* PmT " 2-May-85 20:32")

```
[COND
  ((ACE.CONFIRMIT "Click LEFT to Create a Sequence Window." T)
   (AND ACE.SEQ.WINDOW (CLOSEW ACE.SEQ.WINDOW))
   (SETQ ACE.SEQ.WINDOW (CREATEW (LIST 50 50 (IPLUS ACE.SEQ.WIDTH 8)
                                       (IPLUS ACE.SEQ.HEIGHT 17))
                                ' "Animation Sequence Window" 4))
   (ACE.TELLEM "Position the Sequence Window" T)
   (MOVEW ACE.SEQ.WINDOW))
  ((AND ACE.SEQ.WINDOW (ILEQ ACE.SEQ.WIDTH (WINDOWPROP ACE.SEQ.WINDOW 'WIDTH))
        (ILEQ ACE.SEQ.HEIGHT (WINDOWPROP ACE.SEQ.WINDOW 'HEIGHT))
        (ACE.CONFIRMIT "Click LEFT to Keep Current Window." T))
   (* CLEARW ACE.SEQ.WINDOW)
  )
  (T (ACE.TELLEM "CAUTION: Enter NIL if Unsure at this Stage." T)
    (SETQ ACE.SEQ.WINDOW (EVAL (ACE.ASKEM "Enter the Window: " 'L NIL 120]
  (OR ACE.SEQ.WINDOW (ACE.DEFINE.SEQ.WINDOW))
```

(ACE.FIGURE.OUT.WINDOW

[LAMBDA (REGION/POSITION) (* PmT "22-Apr-85 19:05")

(* This is where all reasoning about which window to use and where offsets should be placed goes.
 Right now (4/20/85) Trillium's just gonna go with positions; but that should
 (?) change)

```

(COND
  (ACE.RUNNING.UNDER.TRILLIUM (DISPLAY.FRAME CURRENT.FRAME)
    (ACE.CREATE.EDITING.BORDER 'INVERT) (* KEEP OFFSETS THE SAME FOR NOW)
  )
  ((POSITIONP REGION/POSITION)
    (SETQ ACE.SEQ.WINDOW.XOFF (CAR REGION/POSITION))
    (SETQ ACE.SEQ.WINDOW.YOFF (CDR REGION/POSITION))
  )
  (T (SETQ ACE.SEQ.WINDOW.XOFF 0)
    (SETQ ACE.SEQ.WINDOW.YOFF 0)
    (ACE.DEFINE.SEQ.WINDOW))
  (ACE.SET.SEQ.CLIP.REGION))

```

(ACE.RETURN.CLOSEST.VERTEX

(* PmT "28-Nov-84 16:15")

```

[LAMBDA (POINT REGION)
  (PROG (NEW.XCOOR NEW.YCOOR)
    [COND
      [(IGREATERP (CAR POINT)
        (SETQ NEW.XCOOR (fetch (REGION RIGHT) of REGION)
        [(ILESSP (CAR POINT)
          (SETQ NEW.XCOOR (fetch (REGION LEFT) of REGION)
          (T (SETQ NEW.XCOOR (CAR POINT)
        ]
      ]
      [(IGREATERP (CDR POINT)
        (SETQ NEW.YCOOR (fetch (REGION TOP) of REGION)
        [(ILESSP (CDR POINT)
          (SETQ NEW.YCOOR (fetch (REGION BOTTOM) of REGION)
          (T (SETQ NEW.YCOOR (CDR POINT)
        ]
      ]
    (RETURN (CONS NEW.XCOOR NEW.YCOOR))
  )

```

(ACE.NEW.SEQ.ASST

(* PmT "23-Jan-85 19:52")

```

[LAMBDA (FIXED MOVE DUM)
  (COND
    ((NULL MOVE)
      (ACE.UPD.CONTROL.WINDOW 'CURSOR FIXED)
      FIXED)
    (T [ACE.UPD.CONTROL.WINDOW 'CURSOR (CONS (ABS (IDIFFERENCE (fetch (POSITION XCOORD) of MOVE)
      (fetch (POSITION XCOORD) of FIXED)))
      (ABS (IDIFFERENCE (fetch (POSITION YCOORD) of MOVE)
      (fetch (POSITION YCOORD) of FIXED))
      MOVE])
  )

```

(ACE.DELAY.FRAME.ASST

(* PmT "21-Dec-84 16:42")

```

[LAMBDA (ITEM MENU MOUSE)
  (COND
    [(FIXP (CADR ITEM))
      (ACE.QUICKDRAW&UPD (LDIFF ACE.CURRENT.SEQUENCE (CDR (NTH ACE.CURRENT.SEQUENCE (CADR ITEM)
      (T NIL]))
  )

```

(ACE.SETUP.CW.CLIPPING.REGIONS

(* PmT "23-Apr-85 13:47")

```

[LAMBDA (MENU)
  (* Sets the clipping region on ACE.CONTROL.WINDOW; There is a menu region, prompt region and status region)

  (PROG (NORMAL ABOVEMENU INFO)
    (WINDOWPROP ACE.CONTROL.WINDOW 'NORMAL.CLIP.REGION (SETQ NORMAL (DSPCLIPPINGREGION NIL
      ACE.CONTROL.WINDOW)))
    [WINDOWPROP ACE.CONTROL.WINDOW 'ABOVMENU.CLIP.REGION (SETQ ABOVEMENU
      (CREATEREGION (fetch (REGION LEFT) of NORMAL)
        (fetch IMAGEHEIGHT of MENU)
        (fetch (REGION WIDTH) of NORMAL)
        (IDIFFERENCE (fetch (REGION HEIGHT)
          of NORMAL)
          (fetch IMAGEHEIGHT of MENU)
        ]
      ]
    [WINDOWPROP ACE.CONTROL.WINDOW 'INFO.CLIP.REGION (SETQ INFO (CREATEREGION (fetch (REGION LEFT)
      of ABOVEMENU)
        (fetch (REGION BOTTOM) of ABOVEMENU)
        130
        (IDIFFERENCE (fetch (REGION HEIGHT)
          of ABOVEMENU)
          5]
      ]
    (WINDOWPROP ACE.CONTROL.WINDOW 'PROMPT.CLIP.REGION (CREATEREGION (IPLUS 3 (fetch (REGION RIGHT)
      of INFO))
        (fetch (REGION BOTTOM) of ABOVEMENU)
        (IDIFFERENCE (fetch (REGION RIGHT)
          of ABOVEMENU)
          (IPLUS 3 (fetch (REGION RIGHT)
            of INFO))
          (IDIFFERENCE (fetch (REGION HEIGHT)
            of ABOVEMENU)
            5)))
      ]
    (DRAWLINE (ADD1 (fetch (REGION RIGHT) of INFO))
      (fetch (REGION BOTTOM) of ABOVEMENU)
      (ADD1 (fetch (REGION RIGHT) of INFO))
    )
  )

```

```

    (fetch (REGION TOP) of ABOVEMENU)
  1
  'PAINT ACE.CONTROL.WINDOW)
(DSPFILL (CREATEREGION 0 0 (fetch IMAGEWIDTH of MENU)
          (fetch IMAGEHEIGHT of MENU))
 38505
'PAINT ACE.CONTROL.WINDOW])

```

(ACE.CHECKSTUFF

```

[LAMBDA (CONDITIONS)                                     (* PmT "26-Oct-84 16:10")
(COND

```

```

  ((EQ CONDITIONS 'SEQ)
   (OR ACE.CURRENT.SEQUENCE (ACE.TELLEM "No Current Sequence defined.  Aborted" T)))
  ((EQ CONDITIONS 'FRAME)
   (OR ACE.CURRENT.FRAME (ACE.TELLEM "No Current Frame.  Aborted" T)))
  ((EQ CONDITIONS 'TAIL)
   (OR ACE.FRAME.TAIL (ACE.TELLEM "Sequence is at End." T))

```

(ACE.UPD.CONTROL.WINDOW

```

[LAMBDA (ITEM VALUE)                                     (* MD "18-Jun-85 16:19")

```

(* This puts info in the status region of the control window; ITEM one of%: CURSOR, FRAME, DEVICE, OPERATION, UPD, T, RESET. VALUE is the value for the ITEM; The ITEMS and VALUEs are stored as WINDOWPROPs on A.C.W)

```

(ACE.MAC.CW.INFO.CLIP (COND
  ((AND (KEYDOWNP 'T)
        (EQ ITEM 'CURSOR))
   (ACE.UPD.CW.MULE 'ACE.CURSOR VALUE))
  ((EQ ITEM 'FRAME)
   [COND
    ((EQ VALUE T)
     (SETQ VALUE (COND
      ((NULL ACE.CURRENT.SEQUENCE)
       'NA)
      ((EQ ACE.CURRENT.SEQUENCE ACE.FRAME.TAIL)
       'START)
      (T (LENGTH (LDIFF ACE.CURRENT.SEQUENCE ACE.FRAME.TAIL))
       (ACE.UPD.CW.MULE 'ACE.FRAME VALUE))
    ))
    ((EQ ITEM 'DEVICE)
     (ACE.UPD.CW.MULE 'ACE.DEVICE VALUE))
    ((EQ ITEM 'OPERATION)
     (ACE.UPD.CW.MULE 'ACE.OPERATION VALUE))
    ((EQ ITEM 'UPD)
     (ACE.UPD.CONTROL.WINDOW2 'FRAME T)
     (ACE.UPD.CW.MULE 'ACE.DEVICE)
     (ACE.UPD.CW.MULE 'ACE.OPERATION 'OK)
     (ACE.UPD.CW.MULE 'ACE.CURSOR 'NA))
    ((EQ ITEM T)
     (ACE.UPD.CW.MULE 'ACE.FRAME)
     (ACE.UPD.CW.MULE 'ACE.DEVICE)
     (ACE.UPD.CW.MULE 'ACE.OPERATION)
     (ACE.UPD.CW.MULE 'ACE.CURSOR))
    ((EQ ITEM 'RESET)
     (ACE.UPD.CONTROL.WINDOW2 'FRAME T)
     (ACE.UPD.CW.MULE 'ACE.DEVICE 'MOUSE)
     (ACE.UPD.CW.MULE 'ACE.OPERATION 'NA)
     (ACE.UPD.CW.MULE 'ACE.CURSOR 'NA))

```

(ACE.UPD.CW.MULE

```

[LAMBDA (ITEM VALUE)                                     (* MD "18-Jun-85 16:49")

```

(* An elaborate WINDOWPROPer. If VALUE is given, it's put on A.C.W as prop ITEM; if VALUE = NIL, returns the current value of ITEM. Also, writes the value in the status region of A.C.W. ITEM one of%: ACE.CURSOR ACE.FRAME ACE.OPERATION ACE.DEVICE. Some restrictions of what VALUE can be (see code); Returns VALUE)

```

(COND
  [(AND (KEYDOWNP 'T)
        (EQ ITEM 'ACE.CURSOR))
   [COND
    ((OR (POSITIONP VALUE)
         (EQ VALUE 'NA))
     (WINDOWPROP ACE.CONTROL.WINDOW ITEM VALUE))
    (T (SETQ VALUE (WINDOWPROP ACE.CONTROL.WINDOW ITEM)
      (ACE.UPD.CLEAR.SET.LINE 4)
    ))
   ]
  ((EQ ITEM 'ACE.FRAME)
   [COND
    ((POSITIONP VALUE)
     (printout ACE.CONTROL.WINDOW " Cursor: " (CAR VALUE)
      " "
      (CDR VALUE)
      .SP 10))
    (T (printout ACE.CONTROL.WINDOW " Cursor: " VALUE .SP 10]

```

```

((OR (FIXP VALUE)
      (EQ VALUE 'START)
      (EQ VALUE 'NA))
  (WINDOWPROP ACE.CONTROL.WINDOW ITEM VALUE))
(T (SETQ VALUE (WINDOWPROP ACE.CONTROL.WINDOW ITEM)
  (ACE.UPD.CLEAR.SET.LINE 1)
  (printout ACE.CONTROL.WINDOW " Frame: " VALUE .SP 20))
  (EQ ITEM 'ACE.OPERATION)
  (COND
    ((NULL VALUE)
      (SETQ VALUE (WINDOWPROP ACE.CONTROL.WINDOW ITEM)))
    (T (WINDOWPROP ACE.CONTROL.WINDOW ITEM)))
  (ACE.UPD.CLEAR.SET.LINE 3)
  (printout ACE.CONTROL.WINDOW " State: " VALUE .SP 20))
  (EQ ITEM 'ACE.DEVICE)
  [COND
    ((OR (EQ VALUE 'MOUSE)
          (EQ VALUE 'MM1201))
      (WINDOWPROP ACE.CONTROL.WINDOW ITEM VALUE))
    (T (SETQ VALUE (WINDOWPROP ACE.CONTROL.WINDOW ITEM)
  (ACE.UPD.CLEAR.SET.LINE 2)
  (printout ACE.CONTROL.WINDOW " Device: " (COND
    ((EQ VALUE 'MOUSE)
      'MOUSE)
    ((EQ VALUE 'MM1201)
      'TABLET)
    (T 'NA))
    .SP 10)))
  VALUE])

```

(ACE.UPD.CLEAR.SET.LINE

(* PmT "17-Dec-84 19:11")

```

[LAMBDA (LINES)
  (MOVETOUPPERLEFT ACE.CONTROL.WINDOW)
  (RELMOVETO 0 (ITIMES (DSPLINEFEED NIL ACE.CONTROL.WINDOW)
    (SUB1 LINES))
    ACE.CONTROL.WINDOW])

```

(ACE.CREATE.CONTROL.MENU

(* MD "26-Jun-85 14:11")

```

[LAMBDA NIL
  (SETQ ACE.CONTROL.MENU (create MENU
    ITEMS _ ' ("Get Sequence" 'I/O%:GET "Fetch a sequence-file")
              ("Edit Frame" 'EDIT "Edits the CURRENT frame")
              ("Run Sequence" 'RUN "Runs the sequence" (SUBITEMS
                ("Loop" 'LOOP "Runs
                  sequence repeatedly
                  until you type a
                  space")
                ("Loop part" 'SUBLOOP
                  "Runs part of the
                  sequence repeatedly
                  ")))
              ("Put Sequence" 'I/O%:PUT "Writes current sequence out to a file")
              ("New Frame" 'FRAME "Adds in another frame AFTER the current one")
              ("Increment Frame" 'INC "Moves forward one frame and displays")
              ("New Sequence" 'NEW "Make a new sequence from scratch")
              ("Delete Frame" 'DEL "Removes CURRENT frame and smoothes over")
              ("Decrement Frame" 'DEC "Goes back one frame")
              ("Reset Sequence" 'RESET "Clears window and resets to start of
                sequence")
              (" Adjust Timing Delays " 'TIME "Manipulate the timing adjustments")
              ("Initialize MM1201 Tablet" 'INIT.MM1201 "Sets up the Tablet for
                use")
              ("Change compression %" 'ACESETTHRESHOLD "Changes the space
                compression factor: 0 to 100 (100 = max compression)")
              ("Change Input Device" 'DEVICE "Select Mouse or Tablet (for now)")
              ("Quit" 'QUIT "Exit ACE; Trillium user's MUST quit when done"))
    CENTERFLG _ T
    MENUCOLUMNS _ 3])

```

(ACE.SEQ.FETCH.WIDTH

(* PmT "22-Apr-85 19:12")

```

[LAMBDA NIL
  (fetch (BITMAP BITMAPWIDTH) of (fetch (ACE.BLIT BITMAP) of (CAR (fetch (ACE.FRAME BLITS) of (CAR
    ACE.CURRENT.SEQUENCE
    ]))

```

(ACE.SEQ.FETCH.HEIGHT

(* PmT "22-Apr-85 19:14")

```

[LAMBDA NIL
  (fetch (BITMAP BITMAPHEIGHT) of (fetch (ACE.BLIT BITMAP) of (CAR (fetch (ACE.FRAME BLITS) of (CAR
    ACE.CURRENT.SEQUENCE
    ]))

```

(ACE.SET.SEQ.CLIP.REGION

```

[LAMBDA (LEFT BOTTOM WIDTH HEIGHT)
  (OR LEFT (SETQ LEFT ACE.SEQ.WINDOW.XOFF))
  (OR BOTTOM (SETQ BOTTOM ACE.SEQ.WINDOW.YOFF))
  (OR WIDTH (SETQ WIDTH ACE.SEQ.WIDTH))
  (OR HEIGHT (SETQ HEIGHT ACE.SEQ.HEIGHT))
  (WINDOWPROP ACE.CONTROL.WINDOW 'SEQUENCE.CLIPPING.REGION (CREATEREGION LEFT BOTTOM WIDTH HEIGHT]))

```

(* "PmT" "17-Apr-85 18:03")

(ACE.ASKEM2

```

[LAMBDA (STRING FLG DEFAULTANSWER TIMELIMIT? SPACES?)

```

(* "PmT" "22-Apr-85 15:56")

(* Like ASKEM but uses whole control window
(use cautiously))

```

  (OR TIMELIMIT? (SETQ TIMELIMIT? 60))
  (COND
    ((EQ FLG T)
     (DSPRESET ACE.CONTROL.WINDOW))
    ((EQ FLG 'L)
     (TERPRI ACE.CONTROL.WINDOW)))
  (MKATOM (PROMPTFORWORD STRING DEFAULTANSWER NIL ACE.CONTROL.WINDOW NIL TIMELIMIT?
    (AND SPACES? (CHARCODE (EOL ESCAPE LF))

```

(ACE.TELLEM2

```

[LAMBDA (STRING FLG)

```

(* "PmT" "19-Dec-84 19:15")

```

  (COND
    ((EQ FLG T)
     (DSPRESET ACE.CONTROL.WINDOW))
    ((EQ FLG 'L)
     (TERPRI ACE.CONTROL.WINDOW)))
  (printout ACE.CONTROL.WINDOW STRING)
  NIL))

```

(ACE.UPD.CONTROL.WINDOW2

```

[LAMBDA (ITEM VALUE)

```

(* "PmT" "19-Dec-84 15:49")

```

  (COND
    ((EQ ITEM 'FRAME)
     [COND
       ((EQ VALUE T)
        (SETQ VALUE (COND
          ((NULL ACE.CURRENT.SEQUENCE)
           'NA)
          ((EQ ACE.CURRENT.SEQUENCE ACE.FRAME.TAIL)
           'START)
          (T (LENGTH (LDIFF ACE.CURRENT.SEQUENCE ACE.FRAME.TAIL]
        (ACE.UPD.CW.MULE 'ACE.FRAME VALUE)))
       (EQ ITEM 'DEVICE)
       (ACE.UPD.CW.MULE 'ACE.DEVICE VALUE))
       (EQ ITEM 'OPERATION)
       (ACE.UPD.CW.MULE 'ACE.OPERATION VALUE))
       (EQ ITEM 'CURSOR)
       (ACE.UPD.CW.MULE 'ACE.CURSOR VALUE))
       (EQ ITEM 'UPD)
       (ACE.UPD.CONTROL.WINDOW2 'FRAME T)
       (ACE.UPD.CW.MULE 'ACE.DEVICE)
       (ACE.UPD.CW.MULE 'ACE.OPERATION 'OK)
       (ACE.UPD.CW.MULE 'ACE.CURSOR 'NA))
       (EQ ITEM T)
       (ACE.UPD.CW.MULE 'ACE.FRAME)
       (ACE.UPD.CW.MULE 'ACE.DEVICE)
       (ACE.UPD.CW.MULE 'ACE.OPERATION)
       (ACE.UPD.CW.MULE 'ACE.CURSOR))
       (EQ ITEM 'RESET)
       (ACE.UPD.CONTROL.WINDOW2 'FRAME T)
       (ACE.UPD.CW.MULE 'ACE.DEVICE 'MOUSE)
       (ACE.UPD.CW.MULE 'ACE.OPERATION 'NA)
       (ACE.UPD.CW.MULE 'ACE.CURSOR 'NA]))
    )

```

(* * The following Macros set up restricting clipping regions)

(DECLARE%: EVAL@COMPILE

(PUTPROPS ACE.MAC.CW.INFO.CLIP MACRO ((FORM)

(RESETLST

```

  [RESETSAVE (PROGN (DSPCLIPPINGREGION (WINDOWPROP ACE.CONTROL.WINDOW
    'INFO.CLIP.REGION)

```

ACE.CONTROL.WINDOW))

(DSPLEFTMARGIN (fetch (REGION LEFT)

of (WINDOWPROP ACE.CONTROL.WINDOW
'INFO.CLIP.REGION))

ACE.CONTROL.WINDOW))

```

  ' (PROGN (DSPCLIPPINGREGION (WINDOWPROP ACE.CONTROL.WINDOW
    'NORMAL.CLIP.REGION)

```

ACE.CONTROL.WINDOW)

```

(DSPLEFTMARGIN (fetch (REGION LEFT)
                      of (WINDOWPROP ACE.CONTROL.WINDOW
                                     'NORMAL.CLIP.REGION))
                ACE.CONTROL.WINDOW]
FORM))

(PUTPROPS ACE.MAC.CW.PROMPT.CLIP MACRO ((FORM)
                                         (RESETLST
                                          [RESETSAVE (PROGN (DSPCLIPPINGREGION (WINDOWPROP
                                                                 ACE.CONTROL.WINDOW
                                                                 'PROMPT.CLIP.REGION)
                                                                 ACE.CONTROL.WINDOW)
                                                              (DSPLEFTMARGIN (fetch (REGION LEFT)
                                                                of (WINDOWPROP
                                                                 ACE.CONTROL.WINDOW
                                                                 'PROMPT.CLIP.REGION)
                                                                )
                                                              ACE.CONTROL.WINDOW))
                                          ' (PROGN (DSPCLIPPINGREGION (WINDOWPROP ACE.CONTROL.WINDOW
                                                                 'NORMAL.CLIP.REGION)
                                                                 ACE.CONTROL.WINDOW)
                                                              (DSPLEFTMARGIN (fetch (REGION LEFT)
                                                                of (WINDOWPROP ACE.CONTROL.WINDOW
                                                                 'NORMAL.CLIP.REGION)
                                                                )
                                                              ACE.CONTROL.WINDOW]
                                          FORM))

(PUTPROPS ACE.MAC.SEQ.CLIP MACRO ((FORM)
                                   (COND
                                    ((WINDOWPROP ACE.CONTROL.WINDOW 'SEQUENCE.CLIPPING.REGION)
                                     (RESETLST
                                      (RESETSAVE (DSPCLIPPINGREGION (WINDOWPROP ACE.CONTROL.WINDOW
                                                                 'SEQUENCE.CLIPPING.REGION)
                                                                 ACE.SEQ.WINDOW)
                                      (LIST 'DSPCLIPPINGREGION (DSPCLIPPINGREGION NIL ACE.SEQ.WINDOW
                                                                )
                                      ACE.SEQ.WINDOW))
                                     FORM))
                                   (T FORM)))


)


(DECLARE%: EVAL@COMPILE


(PUTPROPS ACE.MAC.FETCH.WIDTH MACRO [NIL (fetch (BITMAP BITMAPWIDTH) of (fetch (ACE.BLIT BITMAP)
                                         of (CAR (fetch (ACE.FRAME BLITS)
                                         of (CAR
                                             ACE.CURRENT.SEQUENCE
                                             ]))


(PUTPROPS ACE.MAC.FETCH.HEIGHT MACRO [NIL (fetch (BITMAP BITMAPHEIGHT) of (fetch (ACE.BLIT BITMAP)
                                         of (CAR (fetch (ACE.FRAME BLITS)
                                         of (CAR
                                             ACE.CURRENT.SEQUENCE
                                             ]))

)

(RPAQ ACE.LEFTMOUSE.CURSOR (CURSORCREATE ' 
                               'NIL 8 8))

(RPAQ ACE.MIDDLEMOUSE.CURSOR (CURSORCREATE ' 
                               'NIL 8 8))

(RPAQ ACE.RIGHTMOUSE.CURSOR (CURSORCREATE ' 
                               'NIL 8 8))

(RPAQ ACE.ALLMOUSE.CURSOR (CURSORCREATE ' 
                               'NIL 8 8))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ACE.CONTROL.WINDOW ACE.DIRECTORY ACE.SEQ.WINDOW ACE.SEQ.WIDTH ACE.SEQ.HEIGHT ACE.SEQ.WINDOW.XOFF
             ACE.SEQ.WINDOW.YOFF ACE.CURRENT.SEQUENCE ACE.CURRENT.SEQUENCE.NAME ACE.FRAME.TAIL ACE.CURRENT.FRAME
             ACE.VERTICAL.BLOCK ACE.AREA.THRESHOLD ACE.RUNNING.UNDER.TRILLIUM ACE.LEFTMOUSE.CURSOR
             ACE.MIDDLEMOUSE.CURSOR ACE.RIGHTMOUSE.CURSOR ACE.ALLMOUSE.CURSOR)

)

(* * MENUS IN MAIN)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ACE.CONTROL.MENU ACE.DELAY.MENU ACE.SET.DEVICE.MENU)

```

```

{MEDLEY}<lispusers>ACE>ACE.;1
)

(SETQ ACE.CONTROL.WINDOW NIL)

(SETQ ACE.CONTROL.MENU NIL)

(SETQ ACE.DELAY.MENU NIL)

(SETQ ACE.SET.DEVICE.MENU NIL)

(DECLARE%: DONTVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILEVAR)

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

(RPAQQ ACE-PRIMCOMS
  (( * COMPILER STUFF)
    (VARS ACE.PIXPERWORD ACE.BITMAP.MASK)
    (* LOW LEVEL COMPILER FNS)
    (FNS ACE.COMPILE.FRAME ACE.EXTRACT ACE.SETTHRESHOLD)
    (* REGION MAXING ROUTINES)
    (FNS ACE.MAX.REGIONS ACE.PICK.BEST.REGION ACE.COMPUTE.AREA)
    (* LOW LEVEL BITMAP COMPARISON)
    (FNS ACE.SCAN.BITMAPS ACE.SCAN.PRIMBLOCKS ACE.FETCH.BLOCK)))

  (* * COMPILER STUFF)

(RPAQQ ACE.PIXPERWORD 16)

(RPAQ ACE.BITMAP.MASK (READARRAY-FROM-LIST 16 'SMALLPOSP 0
                                     '(65535 32768 49152 57344 61440 63488 64512 65024 65280 65408 65472 65504 65520
                                     65528 65532 65534 NIL)))

(* * LOW LEVEL COMPILER FNS)

(DEFINEQ

(ACE.COMPILE.FRAME
  [LAMBDA (BM.ORIG BM.CHANGED VERTICAL.BLOCK THRESHOLD)

    (* MJD "28-May-86 16:35")
    (* MAIN ENTRY POINT FOR DIFFERENTIAL BITMAP
    COMPILING)
    (* WARNING%: NO ERROR CHECKING DONE FROM HERE
    DOWN!)
    (* RETURNS A FRAMEPATCH LIST OF ACTUAL BITMAP
    CHANGES)
    (* RETURN FORMAT%: ((BM X . Y) [...]))

    (PROG (CHANGES)
      (SETQ CHANGES (ACE.SCAN.BITMAPS BM.ORIG BM.CHANGED VERTICAL.BLOCK))
      (AND CHANGES (SETQ CHANGES (ACE.MAX.REGIONS CHANGES THRESHOLD)))
      (SETQ CHANGES (ACE.EXTRACT CHANGES BM.CHANGED))
      (ACE.UPD.CONTROL.WINDOW 'OPERATION "DONE")
      (RETURN CHANGES])

(ACE.EXTRACT
  [LAMBDA (REGIONS BITMAP)

    (* MJD "28-May-86 13:12")

    (* TAKES LIST OF REGIONS OF CHANGED AREAS AND MAKES INTO ACTUAL FRAMEPATCH LIST BY EXTRACTING
    FROM NEW BM)

    (PROG (TEMP.BITMAP LEFT BOTTOM WIDTH HEIGHT (FRAMEBLITLIST (CONS)))
      [COND
        ((NULL REGIONS)
          NIL)
        (T (for X in REGIONS do (BLOCK)
          (SETQ LEFT (fetch (REGION LEFT) of X))
          (SETQ BOTTOM (fetch (REGION BOTTOM) of X))
          (SETQ WIDTH (fetch (REGION WIDTH) of X))
          (SETQ HEIGHT (fetch (REGION HEIGHT) of X))
          (SETQ TEMP.BITMAP (BITMAPCREATE WIDTH HEIGHT 1))
          (BITBLT BITMAP LEFT BOTTOM TEMP.BITMAP 0 0 WIDTH HEIGHT 'INPUT 'REPLACE)
          (NCONC1 FRAMEBLITLIST
            (create ACE.BLIT
              BITMAP _ TEMP.BITMAP
              XCOORD _ LEFT
              YCOORD _ BOTTOM)

          (RETURN (CDR FRAMEBLITLIST]))

(ACESETTHRESHOLD

```


)

(DEFINEQ

(* MJD "28-May-86 13:11")

(* RETURNS A LIST OF (REGION REGION |...|))

DONE

(* MJD "28-May-86 13:11")

[COND

[illegible]

```
(ACE.COMPUTE.AREA (UNIONREGIONS (CAR X)
                                (CAR Y))
                  100)))
```

```
(AND (IGREATERP EFFICIENCY (CADDR BEST.SO.FAR))
      (SETQ BEST.SO.FAR (LIST X Y EFFICIENCY))
```

```
(RETURN BEST.SO.FAR])
```

(ACE.COMPUTE.AREA

```
[LAMBDA (REGION EFF)
```

```
(* MJD "28-May-86 13:10")
```

```
(BLOCK)
```

```
(IQUOTIENT (ITIMES (ffetch (REGION WIDTH) of REGION)
                  (ffetch (REGION HEIGHT) of REGION)
```

```
EFF)
```

```
100])
```

```
)
```

```
(* * LOW LEVEL BITMAP COMPARISON)
```

```
(DEFINEQ
```

(ACE.SCAN.BITMAPS

```
[LAMBDA (BM.ORIG BM.NEW BLOCKINGHEIGHT)
```

```
(* PmT "25-Apr-85 15:14")
```

(* Compares BM.ORIG and BM.NEW in one word (ACE.PIXPERWORD bits;
16) by BLOCKINGHEIGHT rectangles. Note masking when get to last word in bitmap and compression of region below
ACE.PIXPERWORD (16); All ARGS required; BM.ORIG and BM.NEW must have the same dimensions!)

```
(* RETURNS A LIST OF TYPE (REGION . 100))
```

```
(PROG [TEMP.ENTRY (BM.WIDTH (ffetch BITMAPWIDTH of BM.ORIG))
```

```
(CHANGED.REGIONS (CONS)
```

```
(RASTERWIDTH (SUB1 (ffetch BITMAPRASTERWIDTH of BM.ORIG)))
```

```
(HEIGHT (SUB1 (ffetch BITMAPHEIGHT of BM.ORIG)))
```

```
(ALLMASK (ELT ACE.BITMAP.MASK 0))
```

```
(PARTIALMASK (ELT ACE.BITMAP.MASK (IMOD (ffetch BITMAPWIDTH of BM.ORIG)
                                         ACE.PIXPERWORD)
```

```
[while (ILESSP Y HEIGHT) bind (Y _ 0)
```

```
do [for HORZ.BLOCK from 0 to RASTERWIDTH
```

```
do (AND [SETQ TEMP.ENTRY (COND
```

```
((EQP HORZ.BLOCK RASTERWIDTH)
```

```
(ACE.SCAN.PRIMBLOCKS BM.ORIG BM.NEW HORZ.BLOCK Y BLOCKINGHEIGHT  
PARTIALMASK))
```

```
(T (ACE.SCAN.PRIMBLOCKS BM.ORIG BM.NEW HORZ.BLOCK Y  
BLOCKINGHEIGHT ALLMASK)
```

```
(NCONC1 CHANGED.REGIONS (CONS [CREATEREGION (ITIMES HORZ.BLOCK ACE.PIXPERWORD)  
(CAR TEMP.ENTRY)
```

```
(IMIN ACE.PIXPERWORD (IDIFFERENCE BM.WIDTH
```

```
(ITIMES HORZ.BLOCK  
ACE.PIXPERWORD  
)))
```

```
(ADD1 (IDIFFERENCE (CDR TEMP.ENTRY)
```

```
(CAR TEMP.ENTRY]
```

```
100]
```

```
(SETQ Y (IPLUS Y BLOCKINGHEIGHT))
```

```
(SETQ BLOCKINGHEIGHT (IMIN BLOCKINGHEIGHT (ADD1 (IDIFFERENCE HEIGHT Y]
```

```
(RETURN (CDR CHANGED.REGIONS]))
```

(ACE.SCAN.PRIMBLOCKS

```
[LAMBDA (BM1 BM2 WORDOFFSET Y0 BLOCKH MASK)
```

```
(* MJD "28-May-86 13:05")
```

(* Does the actual comparison of primitive areas in the two bitmaps BM1 and BM2 ;
WORDOFFSET is the raster word offset; Y0 is the low scanline and
(IPLUS Y0 BLOCKH) is the hi one; MASK is usually \$FFFF, otherwise it is used to ignore extra bits trailing off the end of the
last raster word)

```
(PROG [TEMP1 (MAXY (SUB1 (IPLUS Y0 BLOCKH)
```

```
[SETQ TEMP1 (for Y from Y0 to MAXY thereis (NOT (EQP (LOGAND (LOGXOR (ACE.FETCH.BLOCK BM1 WORDOFFSET Y)  
(ACE.FETCH.BLOCK BM2 WORDOFFSET Y))
```

```
MASK)
```

```
0]
```

```
(RETURN (AND TEMP1 (CONS TEMP1 (for Y from MAXY to TEMP1 by -1
```

```
thereis (NOT (EQP (LOGAND (LOGXOR (ACE.FETCH.BLOCK BM1 WORDOFFSET Y)  
(ACE.FETCH.BLOCK BM2 WORDOFFSET Y))
```

```
MASK)
```

```
0])
```

(ACE.FETCH.BLOCK

```
[LAMBDA (BITMAP WORDOFFSET VERTICAL)
```

```
(* MJD "28-May-86 13:04")
```

(* Nabs a word from bitmap on line VERTICAL with word offset
WORDOFFSET)

```
(BLOCK)
```

```
(\GETBASE (\ADDBASE (ffetch BITMAPBASE of BITMAP)
```

```
(ITIMES (IDIFFERENCE (ffetch BITMAPHEIGHT of BITMAP)
```

```
(ADD1 VERTICAL))
```

```

                (ffetch BITMAPRSTERWIDTH of BITMAP)))
        WORDOFFSET])
)

(RPAQQ ACE-EDITCOMS
[ (* TOP LEVEL EDITING STUFF)
  (FNS ACE.EDIT ACE.EDIT.FRAME ACE.EDIT.SETUP.EDIT.MENU ACEGETREGIONFACTOR ACEROTATEREGION ACESCALEREGION)
  (* LINEART FNS)
  (FNS ACE.EDIT.LINEART ACE.EDIT.LINEART.DRAW ACE.EDIT.LINEART.ADJ ACE.EDIT.LINEART.TRACKLINE)
  (* OTHER EDITING STUFF)
  (FNS ACE.EDIT.MOVE.REGION ACE.EDIT.COMBINE.REGION ACE.EDIT.TEXT ACE.EDIT.TEXTURE.REGION
    ACE.EDIT.TEXTURE.AREA ACE.EDIT.PAINT ACE.FILLWITHTEXTURE ACE.SCANLINESEEDFILL
    ACE.EDIT.CREATE.MENU.TEXTURES ACE.EDIT.PUTDOWN.BITMAP ACE.EDIT.MOVE.REGION.ASST ACEEDITBM
    ACE.READBRUSHSHAPE)
  (* TABLET AND SUPPORT FNS)
  (FNS ACE.EDIT.POINT&CODE ACE.GET.DEVICE.STATE ACE.GET.DEVICE.STATE&CURSOR ACE.EXTRACTBM
    ACE.EDIT.REDRAW.ABITMAP ACE.SCALE.BITMAP ACE.COMPILE.FRAME.ACE ACE.MM1201.INIT ACE.MM1201.POLL
    ACE.MM1201.PROBLEM ACE.EDIT.CLEAR.ALL.MENUS ROTATEBM SIGN TEXTURELINE \TEXTURELINE RS232LOSTCHARFN)
  (MACROS ACE.POPPOS ACE.PUSHPOS)
  (CURSORS ACE.EDIT.LINEART.SQUARE.CURSOR)
  (BITMAPS ACELOGOMAP)
  (VARS (RS232LOSTCHARFN 'RS232LOSTCHARFN))
  (* MENUS FOR ACE-EDIT)
  (GLOBALVARS ACE.CONTROL.WINDOW ACE.EDIT.FRAME.MENU ACE.EDIT.LINEART.ADJ.MENU ACE.EDIT.MOVE.MENU
    ACE.EDIT.TEXT.FONT.MENU ACE.EDIT.TEXT.SIZE.MENU ACE.EDIT.TEXT.FACE.MENU ACE.EDIT.TEXTURE.MENU
    ACE.EDIT.PUTDOWN.MENU)
  (DECLARE%: DONTEVAL@LOAD DOCOPY (P (ACE.EDIT.CLEAR.ALL.MENUS))

  (* * TOP LEVEL EDITING STUFF)

```

(DEFINEQ

(ACE.EDIT

[LAMBDA (FLG)

(* PmT "24-Apr-85 14:51")

(* Entry into frame editing. Reconstructs CURRENT frame, previous
(if any); and successor (if any); Calls ACE.EDIT.FRAME on the reconstructed CURRENT frame
(a bitmap); when editing is complete, recompiles current frame with previous and successor)

```

(PROG (BEFORE.BM PRESENT.BM AFTER.BM)
  (ACE.UPD.CONTROL.WINDOW 'OPERATION "EDIT")
  (COND
    ([NULL (AND (ACE.CHECKSTUFF 'SEQ)
      (ACE.CHECKSTUFF 'FRAME)
      NIL)
    (T [COND
      ((EQ ACE.CURRENT.FRAME ACE.CURRENT.SEQUENCE)
        (SETQ BEFORE.BM NIL))
      (T (SETQ BEFORE.BM (ACE.RECONSTRUCT.FRAME (LDIFF ACE.CURRENT.SEQUENCE ACE.CURRENT.FRAME))
        [COND
          [BEFORE.BM (SETQ PRESENT.BM (ACE.RECONSTRUCT.FRAME (LDIFF ACE.CURRENT.SEQUENCE
            ACE.FRAME.TAIL)
            (T (SETQ PRESENT.BM (fetch (ACE.BLIT BITMAP) of (CAR (fetch (ACE.FRAME BLITS)
              of (CAR ACE.CURRENT.SEQUENCE))
              (NULL ACE.FRAME.TAIL)
              (SETQ AFTER.BM NIL))
              (T (SETQ AFTER.BM (ACE.RECONSTRUCT.FRAME (LDIFF ACE.CURRENT.SEQUENCE (CDR ACE.FRAME.TAIL)
                [COND
                  ((ACE.MAC.SEQ.CLIP (ACE.EDIT.FRAME PRESENT.BM))
                    (COND
                      ((NULL (OR BEFORE.BM AFTER.BM))
                        NIL)
                      ((NULL BEFORE.BM)
                        (replace (ACE.FRAME BLITS) of (CAR ACE.FRAME.TAIL) with (ACE.COMPILE.FRAME.ACE PRESENT.BM
                          AFTER.BM ACE.VERTICAL.BLOCK
                          ACE.AREA.THRESHOLD)))
                      ((NULL AFTER.BM)
                        (replace (ACE.FRAME BLITS) of (CAR ACE.CURRENT.FRAME) with (ACE.COMPILE.FRAME.ACE
                          BEFORE.BM PRESENT.BM
                          ACE.VERTICAL.BLOCK
                          ACE.AREA.THRESHOLD)))
                      (T (replace (ACE.FRAME BLITS) of (CAR ACE.CURRENT.FRAME) with (ACE.COMPILE.FRAME.ACE
                          BEFORE.BM PRESENT.BM
                          ACE.VERTICAL.BLOCK
                          ACE.AREA.THRESHOLD))
                        (replace (ACE.FRAME BLITS) of (CAR ACE.FRAME.TAIL) with (ACE.COMPILE.FRAME.ACE
                          PRESENT.BM AFTER.BM
                          ACE.VERTICAL.BLOCK
                          ACE.AREA.THRESHOLD))
                    (ACE.QUICKDRAW&UPD (LDIFF ACE.CURRENT.SEQUENCE ACE.FRAME.TAIL]))

```

(ACE.EDIT.FRAME

[LAMBDA (ABITMAP)

(* MJD "23-May-86 15:51")

(* Does editing on ABITMAP (this suppose to be a frame; but not necessarily);
Loops through an options menu until QUIT is selected. The BITBLT just redraws the frame as it really is after every editing option)

```
(PROG NIL
  LOOP
    (BITBLT ABITMAP 0 0 ACE.SEQ.WINDOW ACE.SEQ.WINDOW.XOFF ACE.SEQ.WINDOW.YOFF NIL NIL 'INPUT 'REPLACE NIL
      NIL)
    (ACE.UPD.CONTROL.WINDOW 'FRAME T)
    (SELECTQ (MENU (OR ACE.EDIT.FRAME.MENU (ACE.EDIT.SETUP.EDIT.MENU)))
      (NIL NIL)
      (PAINT (ACE.EDIT.PAINT ABITMAP))
      (EDITBM (ACE.EDITBM))
      (LINEART (ACE.EDIT.LINEART ABITMAP))
      (MOVE.REG (ACE.EDIT.MOVE.REGION ABITMAP))
      (COMBINE.REG (ACE.EDIT.COMBINE.REGION ABITMAP))
      (SNAPREG (SNAPW))
      (TEXT (ACE.EDIT.TEXT ABITMAP))
      (TEXTURE.AREA (ACE.EDIT.TEXTURE.AREA ABITMAP))
      (TEXTURE.REG (ACE.EDIT.TEXTURE.REGION ABITMAP))
      (CLEAR (ACE.EDIT.TEXTURE.REGION ABITMAP WHITESHADE))
      (SCALEREG (ACESCALEREGION 'SWEEP))
      (SCALEREGXY (ACESCALEREGION 'XY))
      (SCALEREGX (ACESCALEREGION 'X))
      (SCALEREGY (ACESCALEREGION 'Y))
      (ROTATEREG (ACEROTATEREGION))
      (|2D| (EDIT.SEQ.2D ABITMAP))
      (|3D| (EDIT.SEQ.3D ABITMAP))
      (QUIT (RETURN T))
      (ABORT (AND (ACE.CONFIRMIT "Click LEFT to Ignore Changes." T)
        (RETURN NIL)))
      NIL)
    (GO LOOP])
```

(ACE.EDIT.SETUP.EDIT.MENU

(* MJD "23-May-86 15:42")

```
[LAMBDA NIL
  (SETQ ACE.EDIT.FRAME.MENU (create MENU
    ITEMS _ '(("Paint" 'PAINT "Does standard Paint on frame")
      ("Line art" 'LINEART "Enter lines through various devices")
      ("Text" 'TEXT "Put text into frame")
      ("Edit bits" 'EDITBM "Calls the Bitmap Editor on frame")
      (" " NIL "Just a Spacer")
      ("Move region" 'MOVE.REG "Move a rectangular region in frame")
      ("Combine region" 'COMBINE.REG "Combine any region on screen into
        frame")
      ("Snap region" 'SNAPREG "Save a region in a scratch window")
      ["Scale region" 'SCALEREG "Change size of a region in the frame"
        (SUBITEMS ("To a new region" 'SCALEREG)
          ("In x and y" 'SCALEREGXY)
          ("In x only" 'SCALEREGX)
          ("In y only" 'SCALEREGY)
        )
      ("Rotate region" 'ROTATEREG "Rotates a region any number of
        degrees")
      ("Texture region fill" 'TEXTURE.AREA "Texture any closed area
        within frame")
      ("Texture box fill" 'TEXTURE.REG "Fills a bounded rectangle with
        a texture.")
      ("Clear region" 'CLEAR "Erases a specified Region")
      (" " NIL "Just a Spacer")
      ("Compile frame" 'QUIT "Exits the editor and compiles the
        frame.")
      ("Quit - ABORT" 'ABORT "Stops editor; frame reverts to original
        state."))
    CENTERFLG _ T
    TITLE _ "Edit Options"
    CHANGEOFFSETFLG _ T])
```

(ACEGETREGIONFACTOR

(* MD "19-Jun-85 14:29")

```
[LAMBDA (W H)
  (PROG (TEMPREGION)
    (ACE.TELLEM "Select desired size for new region" T)
    (SETQ TEMPREGION (GETREGION NIL NIL 'ACE.EDIT.MOVE.REGION.ASST NIL))
    (RETURN (CONS (FQUOTIENT (fetch (REGION WIDTH) of TEMPREGION)
      W)
      (FQUOTIENT (fetch (REGION HEIGHT) of TEMPREGION)
      H)))
```

(ACEROTATEREGION

(* MJD "23-May-86 15:15")

```
[LAMBDA (TYPE)
  (PROG (TEMPREGION TEMPBM NEWBM)
    (ACE.TELLEM "Select a Region inside the Sequence" T)
    (SETQ TEMPREGION (GETREGION NIL NIL NIL 'ACE.EDIT.MOVE.REGION.ASST NIL))
    (SETQ TEMPBM (BITMAPCREATE (fetch (REGION WIDTH) of TEMPREGION)
```

```

                (fetch (REGION HEIGHT) of TEMPREGION)
            1))
    (BITBLT ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
      TEMPBM 0 0 NIL NIL 'INPUT 'REPLACE)
    (SETQ NEWBM (BITMAPCREATE (fetch (REGION WIDTH) of TEMPREGION)
      (fetch (REGION HEIGHT) of TEMPREGION)
      1))
    (ROTATEBM TEMPBM NEWBM (RNUMBER "Angle in degrees:"))
    (BITBLT TEMPBM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
      NIL NIL 'INPUT 'ERASE)
    (BITBLT NEWBM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
      NIL NIL 'INPUT 'REPLACE])

```

(ACESCALEREGION

(* MD "19-Jun-85 14:29")

```

[LAMBDA (TYPE)
  (PROG (TEMPREGION TEMPBM NEWBM)
    (ACE.TELLEM "Select a Region inside the Sequence" T)
    (SETQ TEMPREGION (GETREGION NIL NIL NIL 'ACE.EDIT.MOVE.REGION.ASST NIL))
    (SETQ TEMPBM (BITMAPCREATE (fetch (REGION WIDTH) of TEMPREGION)
      (fetch (REGION HEIGHT) of TEMPREGION)
      1))
    (BITBLT ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
      TEMPBM 0 0 NIL NIL 'INPUT 'REPLACE)
    [SETQ NEWBM (COND
      [(EQ TYPE 'SWEEP)
        (ACE.SCALE.BITMAP TEMPBM (ACEGETREGIONFACTOR (fetch (REGION WIDTH) of TEMPREGION)
          (fetch (REGION HEIGHT) of TEMPREGION))
          [(EQ TYPE 'XY)
            (ACE.SCALE.BITMAP TEMPBM (CONS (RNUMBER "%% scale in x")
              (RNUMBER "%% scale in y"))
              ((EQ TYPE 'X)
                (ACE.SCALE.BITMAP TEMPBM (CONS (RNUMBER "%% scale in x")
                  100)))
                ((EQ TYPE 'Y)
                  (ACE.SCALE.BITMAP TEMPBM (CONS 100 (RNUMBER "%% scale in y"))
                    (BITBLT TEMPBM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
                      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
                      NIL NIL 'INPUT 'ERASE)
                    (BITBLT NEWBM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
                      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
                      NIL NIL 'INPUT 'REPLACE))

```

(* * LINEART FNS)

(DEFINEQ

(ACE.EDIT.LINEART

(* MD "19-Jun-85 17:23")
 (* Entry into Lineart on ABITMAP.
 LSHIFT brings up options menu)

```

  (RESETFORM (CURSOR ACE.EDIT.LINEART.SQUARE.CURSOR)
    (PROG (POINT&CODE DEVICE BRUSH MODE)
      (DECLARE (SPECVARS BRUSH MODE))
      (SETQ DEVICE (ACE.UPD.CONTROL.WINDOW 'DEVICE))
      [COND
        ((EQ DEVICE 'MOUSE)
          (ACE.TELLEM "LEFT button Draws (puts Vertices)" T)
          (ACE.TELLEM "MIDDLE Stops drawing; IDLES (Square cursor)" 'L))
        ((EQ DEVICE 'MM1201)
          (ACE.TELLEM "STYLUS Draws (puts down Vertices)" T)
          (ACE.TELLEM "BARREL Stops; IDLES (Square cursor)" 'L)
          (ACE.TELLEM "Right button selects drawing options" 'L)
          (ACE.TELLEM "To exit, use right button, or stylus + barrel (tablet)" 'L))
      LOOP
        (SETQ POINT&CODE (ACE.EDIT.POINT&CODE DEVICE))
        (COND
          ((EQUAL (CAR POINT&CODE)
            'VERTEX)
            (ACE.EDIT.LINEART.DRAW ABITMAP POINT&CODE DEVICE BRUSH MODE))
          ((EQUAL (CAR POINT&CODE)
            'OPTIONS)
            (if (NOT (ACE.EDIT.LINEART.ADJ))
              then (RETURN NIL)))
          ((KEYDOWNP 'RSHIFT)
            (RETURN NIL)))
        (forDuration 30 timerUnits 'MILLISECONDS)
        (GO LOOP])

```

(* So ...LINEART.ADJ sees them.
 Overkill but a good reminder)


```
BRUSH
' INVERT ACE.SEQ.WINDOW])
```

```
)
```

```
(* * OTHER EDITING STUFF)
```

```
(DEFINEQ
```

```
(ACE.EDIT.MOVE.REGION
```

```
[LAMBDA (ABITMAP)
```

```
(* MD "21-Jun-85 13:52")
```

```
(* Moves a region inside the sequence; similar to COMBINE.REGION except the old image may be erased
(thus MOVE); region is confined to sequence by ACE.EDIT.MOVE.REGION.ASST)
```

```
(PROG (TEMP.REGION TEMP.BM)
(ACE.TELLEM "Select a region inside the sequence" T)
(SETQ TEMP.REGION (GETREGION NIL NIL NIL 'ACE.EDIT.MOVE.REGION.ASST NIL))
(SETQ TEMP.BM (BITMAPCREATE (fetch (REGION WIDTH) of TEMP.REGION)
(fetch (REGION HEIGHT) of TEMP.REGION)
1))
(BITBLT ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMP.REGION))
(ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMP.REGION))
TEMP.BM 0 0 NIL NIL 'INPUT 'REPLACE)
(SELECTQ [MENU (OR ACE.EDIT.MOVE.MENU (SETQ ACE.EDIT.MOVE.MENU
(create MENU
ITEMS _ ' ((Erase 'ERASE "NAND the old image")
(Invert 'XOR "XOR the old image with itself")
(Nothing NIL "Just leave the old image as is")
)
CENTERFLG _ T
TITLE _ "Do What with Old Image?"
CHANGEOFFSETFLG _ T]
(NIL NIL)
(ERASE (BITBLT TEMP.BM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMP.REGION))
(ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMP.REGION))
NIL NIL 'INPUT 'ERASE))
(XOR (BITBLT TEMP.BM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMP.REGION))
(ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMP.REGION))
NIL NIL 'INPUT 'INVERT))
NIL)
(ACE.EDIT.PUTDOWN.BITMAP TEMP.BM ABITMAP])
```

```
(ACE.EDIT.COMBINE.REGION
```

```
[LAMBDA (ABITMAP)
```

```
(* PmT "25-Apr-85 15:22")
```

```
(* Grabs a region from the screen; makes it into a bitmap, then pastes it into the current frame;
see ACE.EDIT.PUTDOWN.BITMAP)
```

```
(PROG (TEMP.BM TEMP.REGION)
(ACE.TELLEM "Select a Region from Screen" T)
(SETQ TEMP.REGION (GETREGION))
(SETQ TEMP.BM (BITMAPCREATE (fetch (REGION WIDTH) of TEMP.REGION)
(fetch (REGION HEIGHT) of TEMP.REGION)
1))
(BITBLT (SCREENBITMAP)
(fetch (REGION LEFT) of TEMP.REGION)
(fetch (REGION BOTTOM) of TEMP.REGION)
TEMP.BM 0 0 NIL NIL 'INPUT 'REPLACE)
(ACE.EDIT.PUTDOWN.BITMAP TEMP.BM ABITMAP])
```

```
(ACE.EDIT.TEXT
```

```
[LAMBDA (ABITMAP)
```

```
(* PmT "25-Apr-85 17:57")
```

```
(* Puts text into current frame; actually, puts text into ABITMAP
```

```
and the sequence window)
```

```
(PROG ((FONT.FAMILY NIL)
(FONT.SIZE NIL)
(FONT.FACE NIL)
(DUMB.DSP.FOR.BM (DSPCREATE ABITMAP))
BMX BMY USERSTRING POSITION POINT&CODE FONT)
(GETMOUSESTATE)
(SETQ POSITION (CONS LASTMOUSEX LASTMOUSEY))
(SETQ FONT.FAMILY (MENU (OR ACE.EDIT.TEXT.FONT.MENU (SETQ ACE.EDIT.TEXT.FONT.MENU
```

```
(* This so MENUs stay in one (the same) place!)
```

```
(create MENU
ITEMS _ ' ((Classic 'CLASSIC)
(Cream 'CREAM)
(Gacha 'GACHA)
(Helvetica 'HELVETICA)
(Modern 'MODERN)
(TimesRoman 'TIMESROMAN)
(" " NIL)
(* Same * " 'SAME "Use same
descriptor as last time")
(* Other * " 'OTHER))
```

```

TITLE _ "Select a Font Family"
CENTERFLG _ T))

        POSITION))
(COND
  [(EQ FONT.FAMILY 'SAME)
   (SETQ FONT.FAMILY (FONTPROP (DSPFONT NIL ACE.SEQ.WINDOW)
                                'FAMILY))
   (SETQ FONT.SIZE (FONTPROP (DSPFONT NIL ACE.SEQ.WINDOW)
                              'SIZE))
   (SETQ FONT.FACE (FONTPROP (DSPFONT NIL ACE.SEQ.WINDOW)
                              'FACE)]
  [(EQ FONT.FAMILY 'OTHER)
   (SETQ FONT.FAMILY (ACE.ASKEM "Enter a Font Family: " T))
   (NULL FONT.FAMILY)
   (RETURN NIL))
[COND
  (FONT.SIZE NIL)
  [(NEQ 'OTHER (SETQ FONT.SIZE
                    (MENU (OR ACE.EDIT.TEXT.SIZE.MENU
                           (SETQ ACE.EDIT.TEXT.SIZE.MENU
                                (create MENU
                                  ITEMS _ '(6 8 10 12 14 16 18 24 36 ("* Other *" 'OTHER))
                                  TITLE _ "Select Font Size"
                                  CENTERFLG _ T
                                  MENUROWS _ 2)))
                           POSITION])
        (T (SETQ FONT.SIZE (ACE.ASKEM "Enter a Font Size: " T)
[COND
  (FONT.FACE NIL)
  [(NEQ 'OTHER (SETQ FONT.FACE
                    (MENU (OR ACE.EDIT.TEXT.FACE.MENU
                           (SETQ ACE.EDIT.TEXT.FACE.MENU
                                (create MENU
                                  ITEMS _ '(("Standard" 'STANDARD "Normal font face")
                                             ("Italic" 'ITALIC "A standard face in italic")
                                             ("Expanded" 'MRE)
                                             ("Bold" 'BOLD "A heavy bold face")
                                             ("Bold Italic" 'BIR)
                                             ("Compressed" 'MRC)
                                             (" " NIL)
                                             ("* Other *" 'OTHER)
                                             (" " NIL))
                                  TITLE _ "Select a Font Face"
                                  CENTERFLG _ T
                                  MENUROWS _ 3
                                  MENCOLUMNS _ 3)))
                           POSITION])
        (T (SETQ FONT.FACE (ACE.ASKEM "Enter a Font Face: " T)
[COND
  [(SETQ FONT (FONTCREATE FONT.FAMILY FONT.SIZE FONT.FACE NIL NIL T))
   (ACE.TELLEM "Ready to start Entering Text." T)
   (ACE.TELLEM "Position Mouse and Click LEFT." 'L)
   (until (EQ [CAR (SETQ POINT&CODE (ACE.EDIT.POINT&CODE 'MOUSE]
                                'VERTEX)
           do)
    (ACE.TELLEM "A RETURN Stops Text." T)
    (SETQ BMX (CADR POINT&CODE))
    (SETQ BMY (CDDR POINT&CODE))
    (MOVETO (ACE.MT.SEQX.AWX BMX)
            (ACE.MT.SEQY.AWY BMY)
            ACE.SEQ.WINDOW)
    (MOVETO BMX BMY DUMB.DSP.FOR.BM)
    [RESETFORM (DSPFONT FONT ACE.SEQ.WINDOW)
      (SETQ USERSTRING (PROMPTFORWARD NIL NIL NIL ACE.SEQ.WINDOW NIL NIL (CHARCODE (EOL ESCAPE LF)
                                                                                     ]
    (RESETFORM (DSPFONT FONT DUMB.DSP.FOR.BM)
      (AND USERSTRING (PRIN1 USERSTRING DUMB.DSP.FOR.BM)
    (T (ACE.TELLEM "Can't find any such Font. Aborted." T])

```

(ACE.EDIT.TEXTURE.REGION

[LAMBDA (ABITMAP SHADE)

(* PmT "25-Apr-85 15:33")

(* Textures a region in ABITMAP; SHADE is optional; if not given, user is asked for a shade)

```

(PROG (OPERATION SHADEREGION)
[COND
  (SHADE NIL)
  [(NEQ 'OTHER (SETQ SHADE (MENU (OR ACE.EDIT.TEXTURE.MENU (SETQ ACE.EDIT.TEXTURE.MENU (
                                                                                     ACE.EDIT.CREATE.MENU.TEXTURES
                                                                                     ]
        (T (SETQ SHADE (EDITSHADE)
          (OR SHADE (RETURN NIL))
          (ACE.TELLEM "Select a Region in the Sequence" T)
          (SETQ SHADEREGION (GETREGION))
          [SETQ OPERATION (COND
            ((EQ SHADE WHITESHADE)

```



```

      'REPLACE)
      (MENU (OR ACE.EDIT.PUTDOWN.MENU (SETQ ACE.EDIT.PUTDOWN.MENU
        (create MENU
          ITEMS _ ' ("Paint" 'PAINT "Ors with
                    Frame")
                    ("Replace" 'REPLACE "Puts onto
                    Frame")
                    ("Invert" 'INVERT "XORs with
                    Frame")
                    ("Erase" 'ERASE "NANDs with
                    Frame"))
          CENTERFLG _ T
          TITLE _ "Select a Drawing Mode"]
      (replace (REGION LEFT) of SHADEREGION with (ACE.MT.SCRX.AWX (fetch (REGION LEFT) of SHADEREGION)))
      (replace (REGION BOTTOM) of SHADEREGION with (ACE.MT.SCRY.AWY (fetch (REGION BOTTOM) of SHADEREGION)))
      (BITBLT NIL NIL NIL ACE.SEQ.WINDOW (fetch (REGION LEFT) of SHADEREGION)
        (fetch (REGION BOTTOM) of SHADEREGION)
        (fetch (REGION WIDTH) of SHADEREGION)
        (fetch (REGION HEIGHT) of SHADEREGION)
        'TEXTURE OPERATION SHADE (ACE.MT.SEQ.AW.REGION))
      (BITBLT NIL NIL NIL ABITMAP (ACE.MT.AWX.SEQX (fetch (REGION LEFT) of SHADEREGION))
        (ACE.MT.AWY.SEQY (fetch (REGION BOTTOM) of SHADEREGION))
        (fetch (REGION WIDTH) of SHADEREGION)
        (fetch (REGION HEIGHT) of SHADEREGION)
        'TEXTURE OPERATION SHADE))

```

(ACE.EDIT.TEXTURE.AREA

[LAMBDA (ABITMAP SHADE)

(* PmT " 2-May-85 19:47")

(* Does an area flood on a bounded region. ABITMAP is the frame (required); SHADE is an optional shade arg (SMALLP); Works by making a bitmap copy of a user selected region. Then flooding whatever area is enclosed with a user seed point.)

```

(PROG (BOUNDING.REGION SEED.POINT TEMP.BM)
  (ACE.TELLEM "Select a Maximum Bounding Region." T)
  (SETQ BOUNDING.REGION (GETREGION NIL NIL NIL 'ACE.EDIT.MOVE.REGION.ASST NIL))
  (ACE.TELLEM "Select a Starting Point Inside the Region." T)
  (SETQ SEED.POINT (GETPOSITION))
  (ACE.TELLEM "Select a Fill Shade." T)
  [COND
    (SHADE)
    [(NEQ 'OTHER (SETQ SHADE (MENU (OR ACE.EDIT.TEXTURE.MENU (SETQ ACE.EDIT.TEXTURE.MENU (
      ACE.EDIT.CREATE.MENU.TEXTURES
    ]
    (T (SETQ SHADE (EDITSHADE)
  [COND
    ((NOT (INSIDEP BOUNDING.REGION SEED.POINT))
      (ACE.TELLEM "Your Seed Point lies Outside of your Region." T)
      (ACE.TELLEM "Texture Area Fill Aborted." 'L)
      (RETURN NIL))
    ((NULL SHADE)
      (ACE.TELLEM "No Shade selected. Aborted." T)
      (RETURN NIL))
    (T (SETQ SEED.POINT (CONS (IDIFFERENCE (CAR SEED.POINT)
      (fetch (REGION LEFT) of BOUNDING.REGION))
      (IDIFFERENCE (CDR SEED.POINT)
      (fetch (REGION BOTTOM) of BOUNDING.REGION)
      1))
      (SETQ TEMP.BM (BITMAPCREATE (fetch (REGION WIDTH) of BOUNDING.REGION)
      (fetch (REGION HEIGHT) of BOUNDING.REGION)
      1))
      (BITBLT ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of BOUNDING.REGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of BOUNDING.REGION))
      TEMP.BM 0 0 NIL NIL 'INPUT 'REPLACE)
      (ACE.TELLEM "Texturing Area..." T)
      [RESETFORM (CURSOR WAITINGCURSOR)
        (ACE.FILLWITHTEXTURE TEMP.BM SHADE (CAR SEED.POINT)
          (CDR SEED.POINT)
          (LOGXOR 1 (BITMAPBIT TEMP.BM (CAR SEED.POINT)
            (CDR SEED.POINT)
            (CDR SEED.POINT)
            (CDR SEED.POINT))
          (BITBLT TEMP.BM 0 0 ACE.SEQ.WINDOW (ACE.MT.SCRX.AWX (fetch (REGION LEFT) of BOUNDING.REGION))
            (ACE.MT.SCRY.AWY (fetch (REGION BOTTOM) of BOUNDING.REGION))
            NIL NIL 'INPUT 'REPLACE NIL NIL)
      (COND
        ((ACE.CONFIRMIT "Click LEFT to Confirm Fill as Shown." T)
          (BITBLT TEMP.BM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of BOUNDING.REGION))
            (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of BOUNDING.REGION))
            NIL NIL 'INPUT 'REPLACE NIL NIL)
          (T (ACE.TELLEM "Texture Area Fill Aborted." T]))

```

(ACE.EDIT.PAINT

[LAMBDA (ABITMAP)

; Edited 3-Sep-88 15:00 by masinter
 (* Hacked from rrb "18-OCT-83 18:37")
 (* Paint on current frame with either Mouse or Tablet)

(* should make sure cursor has moved or a button has change before proceeding with the inner loop.)

(* has some of the stuff to allow the brush to be an arbitrary bitmap but not all.)

```
(ACE.TELLEM "Left (Mouse), Stylus (Tablet) Paints." T)
(ACE.TELLEM "Middle (Mouse), Barrel (Tablet) Erases." 'L)
(ACE.TELLEM "Right (Mouse) or Left Shift for Menu." 'L)
(ACE.TELLEM "To Quit, select Quit from Menu." 'L)
(BITBLT ABITMAP 0 0 ACE.SEQ.WINDOW ACE.SEQ.WINDOW.XOFF ACE.SEQ.WINDOW.YOFF NIL NIL 'INPUT 'REPLACE NIL NIL)
(PROG (OLDX OLDY (NEWBRUSHQ T)
      BRUSH WINDOW (ORIG.CURSOR (CURSOR))
      (ORIG.REGION (DSPCLIPPINGREGION NIL ACE.SEQ.WINDOW)))
  (SETQ WINDOW ACE.SEQ.WINDOW)
  (SETQ WINDOW (\INSUREWINDOW WINDOW))
  (RESETLST
    (RESETSAVE NIL (LIST 'CURSOR (CURSOR)))
    (PROG (DS HOTX HOTY)
      (TOTOPW WINDOW)
      (* look for a previously stored brush.)
      [COND
        ((SETQ BRUSH (WINDOWPROP WINDOW 'PAINTBRUSH))
          (SETQ PAINTCOMMANDMODE (CAR BRUSH))
          (SETQ PAINTCOMMANDSHADE (CADR BRUSH))
          (SETQ PAINTCOMMANDBRUSH (CADDR BRUSH))
          (SETQ DS (WINDOWPROP WINDOW 'DSP))
        )
      BRUSHLP
      [AND NEWBRUSHQ (SETQ BRUSH (COND
        ((BITMAPP PAINTCOMMANDBRUSH))
        ((EQ (CAR PAINTCOMMANDBRUSH)
              'other)
          (SETQ NEWBRUSHQ NIL)
          (ACE.EXTRACTBM))
        (T (\GETBRUSH PAINTCOMMANDBRUSH))
      )
      [AND (BITMAPP PAINTCOMMANDBRUSH)
        (SETQ PAINTCOMMANDBRUSH ' (SQUARE 2] (* clear cursor)
        (BITBLT NIL NIL NIL (CURSORBITMAP)
          0 0 16 16 'TEXTURE 'REPLACE WHITESHADE)
          (* put lower left part of brush shape in cursor)
          (* BITBLT BRUSH 0 0 (SCREENBITMAP) 0 0 NIL NIL
            (QUOTE INPUT) (QUOTE REPLACE))
          (* set the hot spot to the middle of the brush.)
        ]
      [CURSORHOTSPOT (create POSITION
        XCOORD _ (SETQ HOTX (IDIFFERENCE (IMIN (fetch BITMAPWIDTH of BRUSH)
          16)
          2))
        YCOORD _ (SETQ HOTY (IDIFFERENCE (IMIN (fetch BITMAPHEIGHT of BRUSH)
          16)
          2]
      )
    PAINTLP
    (ACE.GET.DEVICE.STATE&CURSOR)
    [COND
      ((KEYDOWNP 'RSHIFT)
        (RETURN))
      ((OR (LASTMOUSESTATE RIGHT)
        (KEYDOWNP 'LSHIFT))
        (COND
          ((OR (INSIDE? (DSPCLIPPINGREGION NIL DS)
            (LASTMOUSEX DS)
            (LASTMOUSEY DS))
            (NOT (WHICHW LASTMOUSEX LASTMOUSEY)))
            (* inside the interior, give command menu)
          )
          (SELECTQ [MENU (COND
            ((type? MENU PAINTCOMMANDMENU)
              PAINTCOMMANDMENU)
            (T (SETQ PAINTCOMMANDMENU (create MENU
              ITEMS _
              ' ((HardCopy 'HARDCOPY "Makes a
                press file of the window
                and prints it")
                (SetMode 'MODE "Allows
                specification of how new
                bits are merged")
                (SetShade 'SHADE "Allows
                specification of new
                shade.")
                (SetShape 'SHAPE "Allows
                specification of brush
                shape")
                (SetSize 'SIZE "Allows
                specification of the
                brush size")
                (QUIT 'QUIT "Exits painting
                mode"))
              CHANGEOFFSETFLG _ T]
            (SHADE (SETQ PAINTCOMMANDSHADE (OR (PAINTW.READBRUSHSHADE)
              PAINTCOMMANDSHADE))
              (GO BRUSHLP))
            (MODE (SETQ PAINTCOMMANDMODE (OR (PAINTW.READMODE)
              PAINTCOMMANDMODE))
          )
        )
      )
    )
  )

```

```

(GO BRUSHLP))
(SHAPE (RPLACA PAINTCOMMANDBRUSH (OR (ACE.READBRUSHSHAPE)
(CAR PAINTCOMMANDBRUSH)))
      (SETQ NEWBRUSHQ T)
      (GO BRUSHLP))
(SIZE (RPLACA (CDR PAINTCOMMANDBRUSH)
      (OR (PAINTW.READBRUSHSIZE)
      (CADR PAINTCOMMANDBRUSH)))
      (GO BRUSHLP))
(QUIT (WINDOWPROP ACE.SEQ.WINDOW 'PAINTBRUSH (LIST PAINTCOMMANDMODE
PAINTCOMMANDSHADE
(OR (BITMAPP BRUSH)
PAINTCOMMANDBRUSH)))
      (SETQ PAINTCOMMANDBRUSH ' (ROUND 16))
      (RETURN))
(HARDCOPY (RESETFORM (TTYDISPLAYSTREAM PROMPTWINDOW)
      (HARDCOPYW WINDOW)))
NIL))
(T
  NIL))) (* do NOT do the window menu)
[ (AND (LASTMOUSESTATE LEFT)
      (OR (EQ PAINTCOMMANDMODE 'REPLACE)
      (NEQ PAINTCOMMANDSHADE BLACKSHADE)))
  (* painting in grey is slightly harder.)
  (COND
    ((EQ PAINTCOMMANDMODE 'REPLACE) (* erase what is there now)
     (BITBLT BRUSH 0 0 DS (IDIFFERENCE (LASTMOUSEX DS)
                                         HOTX)
              (IDIFFERENCE (LASTMOUSEY DS)
                           HOTY)
              NIL NIL 'INPUT 'ERASE) (* put in grey)
     (BITBLT BRUSH 0 0 DS (IDIFFERENCE (LASTMOUSEX DS)
                                         HOTX)
              (IDIFFERENCE (LASTMOUSEY DS)
                           HOTY)
              NIL NIL 'MERGE 'PAINT PAINTCOMMANDSHADE))
    (T (BITBLT BRUSH 0 0 DS (IDIFFERENCE (LASTMOUSEX DS)
                                         HOTX)
              (IDIFFERENCE (LASTMOUSEY DS)
                           HOTY)
              NIL NIL 'MERGE PAINTCOMMANDMODE PAINTCOMMANDSHADE]
      [(LASTMOUSESTATE (OR MIDDLE LEFT))
       (BITBLT BRUSH 0 0 DS (IDIFFERENCE (LASTMOUSEX DS)
                                         HOTX)
                           (IDIFFERENCE (LASTMOUSEY DS)
                                         HOTY)
                           NIL NIL 'INPUT (COND
                                         ((LASTMOUSESTATE MIDDLE)
                                          'ERASE)
                                         (T PAINTCOMMANDMODE)
                                         (* Idle -
                                         just mark brush loc.)
                                         (BITBLT BRUSH 0 0 DS (SETQ OLDX (IDIFFERENCE (LASTMOUSEX DS)
                                         HOTX))
                                         (SETQ OLDY (IDIFFERENCE (LASTMOUSEY DS)
                                         HOTY))
                                         NIL NIL 'INPUT 'INVERT)
                                         (DISMISS 2)
                                         (BITBLT BRUSH 0 0 DS OLDX OLDY NIL NIL 'INPUT 'INVERT]
      (GO PAINTLP))
      (WINDOWPROP WINDOW 'PAINTBRUSH (LIST PAINTCOMMANDMODE PAINTCOMMANDSHADE (COPY PAINTCOMMANDBRUSH)))
      (BITBLT WINDOW ACE.SEQ.WINDOW.XOFF ACE.SEQ.WINDOW.YOFF ABITMAP 0 0 ACE.SEQ.WIDTH ACE.SEQ.HEIGHT
        'INPUT
        'REPLACE NIL NIL))
      (WINDOWPROP ACE.SEQ.WINDOW 'PAINTBRUSH (LIST PAINTCOMMANDMODE PAINTCOMMANDSHADE (OR (BITMAPP BRUSH)
PAINTCOMMANDBRUSH)
)

```

(ACE.FILLWITHTEXTURE

[LAMBDA (BitmapOrWindow Texture X Y BoundaryValue)

(* PmT " 2-May-85 18:45")

(* hdj " 5-Mar-85 15:53")

(* This code created by Herb Jellinek and Kelly Roach. Renamed for organizational purposes.
 Paints in TEXTURE into supplied BitmapOrWindow (Only a bitmap here);
 BoundaryValue should be either a 1 or 0 (No color bitmaps or anything fancy for now);
 X and Y specify the seed point for the fill)

```

(PROG ((COPYBM (COPYALL BitmapOrWindow)))
  (ACE.SCANLINESEEDFILL COPYBM X Y BoundaryValue BoundaryValue)
  (BITBLT BitmapOrWindow 0 0 COPYBM 0 0 NIL NIL 'INPUT 'INVERT)
  (BITBLT COPYBM 0 0 COPYBM 0 0 NIL NIL 'MERGE 'TEXTURE Texture)
  (BITBLT COPYBM 0 0 BitmapOrWindow 0 0 NIL NIL 'INPUT 'PAINT)
  BitmapOrWindow])

```

(ACE.SCANLINESEEDFILL

```

[LAMBDA (BitmapOrWindow X Y BoundaryValue FillValue)
  (*PmT "2-May-85 20:19")
  (* hdj "30-Jan-85 15:01")
  (* This code created by Herb Jellinek and Kelly Roach.
  Renamed for organizational purposes)

  (PROG (Xcoord Ycoord STACK SaveX SaveY XLeft XRight XMax YMax)
    [if (BITMAPP BitmapOrWindow)
      then (SETQ XMax (SUB1 (BITMAPWIDTH BitmapOrWindow)))
            (SETQ YMax (SUB1 (BITMAPHEIGHT BitmapOrWindow)))
      else [SETQ XMax (SUB1 (WINDOWPROP BitmapOrWindow 'WIDTH])
            (SETQ YMax (SUB1 (WINDOWPROP BitmapOrWindow 'HEIGHT])
            (* "initialize stack")

    (ACE.PUSHPOS X Y STACK)
    (while STACK do
      (* get seed pixel and set to new value)
      (ACE.POPPOS STACK SaveX SaveY)
      (BITMAPBIT BitmapOrWindow SaveX SaveY FillValue)
      (* fill span to right of seed pixel)

      (SETQ XRight XMax)
      (for Xcoord from (ADD1 SaveX) while (ILEQ Xcoord XMax)
        do (if (NEQ (BITMAPBIT BitmapOrWindow Xcoord SaveY)
                    BoundaryValue)
          then (BITMAPBIT BitmapOrWindow Xcoord SaveY FillValue)
          else (* save the extreme right pixel)
              (SETQ XRight (SUB1 Xcoord))
              (* fill span to left of seed pixel)
              (RETURN)))

      (SETQ XLeft 0)
      (for Xcoord from (SUB1 SaveX) by -1 while (IGEQ Xcoord 0)
        do (if (NEQ (BITMAPBIT BitmapOrWindow Xcoord SaveY)
                    BoundaryValue)
          then (BITMAPBIT BitmapOrWindow Xcoord SaveY FillValue)
          else (* save the extreme left pixel)
              (SETQ XLeft (ADD1 Xcoord))
              (RETURN)))
      (* Push seed points for scan line above.
      *)

    [COND
      ((ILESSP SaveY YMax)
        (SETQ Ycoord (ADD1 SaveY))
        (for Xcoord from XLeft to XRight
          when [AND (NEQ (BITMAPBIT BitmapOrWindow Xcoord Ycoord)
                        BoundaryValue)
                    (OR (EQ Xcoord XRight)
                        (OR (EQ (BITMAPBIT BitmapOrWindow (ADD1 Xcoord)
                                                              Ycoord)
                          BoundaryValue)
                          (EQ (BITMAPBIT BitmapOrWindow (ADD1 Xcoord)
                                                              Ycoord)
                              FillValue))]
            do (ACE.PUSHPOS Xcoord Ycoord STACK)
            (* Push seed points for scan line below.
            *)

      (COND
        ((IGREATERP SaveY 0)
          (SETQ Ycoord (SUB1 SaveY))
          (for Xcoord from XLeft to XRight
            when [AND (NEQ (BITMAPBIT BitmapOrWindow Xcoord Ycoord)
                          BoundaryValue)
                      (OR (EQ Xcoord XRight)
                          (OR (EQ (BITMAPBIT BitmapOrWindow (ADD1 Xcoord)
                                                                Ycoord)
                            BoundaryValue)
                            (EQ (BITMAPBIT BitmapOrWindow (ADD1 Xcoord)
                                                                Ycoord)
                                FillValue))]
              do (ACE.PUSHPOS Xcoord Ycoord STACK]))

  (ACE.EDIT.CREATE.MENU.TEXTURES
    [LAMBDA NIL
      (*PmT "25-Apr-85 15:27")
      (* Creates a textures menu for TEXTURE.REGION and FILL
      routines)

      (PROG (TEMP.BM)
        (RETURN (create MENU
          ITEMS _
          (NCONC1 (for TEXTURE
            in ' (65535 34850 43605 63624 42405 4080 64250 26214 65488 34925 15 4680 33825
              1185 1 3784 3591)
            collect (PROGN (BITBLT NIL NIL NIL (SETQ TEMP.BM (BITMAPCREATE 36 36 1))
              4 4 28 28 'TEXTURE 'REPLACE TEXTURE NIL)
              (LIST TEMP.BM TEXTURE)))
            ' ("* Other *" 'OTHER "Make your own shade"))
          TITLE _ "Texture Menu"
          CENTERFLG _ T
          MENUROWS _ 3]))

  (ACE.EDIT.PUTDOWN.BITMAP
    [LAMBDA (IMAGE.BM TARGET.BM)
      (* MD "21-Jun-85 13:56")

```

(* Pastes IMAGE.BM onto TARGET.BM; IMAGE.BM is tied to mouse until left click;
asks user how to combine the two)

```
(PROG (MODE POINT)
  (ACE.TELLEM "Click Left to paste down image." T)
  (SETQ POINT (PROG (OLD.X OLD.Y)
    LOOP
      (BITBLT IMAGE.BM 0 0 ACE.SEQ.WINDOW (SETQ OLD.X (LASTMOUSEX ACE.SEQ.WINDOW))
        (SETQ OLD.Y (LASTMOUSEY ACE.SEQ.WINDOW))
        NIL NIL 'INPUT 'INVERT NIL (ACE.MT.SEQ.AW.REGION))
      (GETMOUSESTATE)
      (ACE.UPD.CONTROL.WINDOW 'CURSOR (CONS (ACE.MT.SCRX.SEQX LASTMOUSEX)
        (ACE.MT.SCRY.SEQY LASTMOUSEY)))
      [COND
        [(LASTMOUSESTATE LEFT)
          (do (GETMOUSESTATE) until (LASTMOUSESTATE UP))
          (BITBLT IMAGE.BM 0 0 ACE.SEQ.WINDOW OLD.X OLD.Y NIL NIL 'INPUT 'INVERT NIL
            (ACE.MT.SEQ.AW.REGION))
          (RETURN (CONS (LASTMOUSEX ACE.SEQ.WINDOW)
            (LASTMOUSEY ACE.SEQ.WINDOW))]
          (T (BITBLT IMAGE.BM 0 0 ACE.SEQ.WINDOW OLD.X OLD.Y NIL NIL 'INPUT 'INVERT NIL
            (ACE.MT.SEQ.AW.REGION))
          (GO LOOP)))
      (SETQ MODE (MENU (OR ACE.EDIT.PUTDOWN.MENU (SETQ ACE.EDIT.PUTDOWN.MENU
        (create MENU
          ITEMS _ ' ("Paint" 'PAINT "ORs with Frame")
            ("Replace" 'REPLACE "Puts onto Frame")
            ("Invert" 'INVERT "XORs with Frame")
            ("Erase" 'ERASE "NANDs with Frame"))
          CENTERFLG _ T
          TITLE _ "Select a Drawing Mode"
          CHANGEOFFSETFLG _ T]
        (BITBLT IMAGE.BM 0 0 ACE.SEQ.WINDOW (CAR POINT)
          (CDR POINT)
          NIL NIL 'INPUT MODE NIL (ACE.MT.SEQ.AW.REGION))
        (BITBLT IMAGE.BM 0 0 TARGET.BM (ACE.MT.AWX.SEQX (CAR POINT))
          (ACE.MT.AWY.SEQY (CDR POINT))
          NIL NIL 'INPUT MODE NIL]))
```

(ACE.EDIT.MOVE.REGION.ASST

(* PmT "23-Jan-85 20:03")

```
[LAMBDA (FIXED MOVE EXTRA)
  (COND
    [(NULL MOVE)
      (ACE.NEW.SEQ.ASST (CONS (ACE.MT.SCRX.SEQX (CAR FIXED))
        (ACE.MT.SCRY.SEQY (CDR FIXED)))
      MOVE EXTRA)
    (COND
      ((INSIDEP (ACE.MT.SEQ.SCR.REGION)
        FIXED)
        (T (ACE.RETURN.CLOSEST.VERTEX FIXED (ACE.MT.SEQ.SCR.REGION)]
      (T (ACE.NEW.SEQ.ASST FIXED MOVE EXTRA)
        (COND
          ((INSIDEP (ACE.MT.SEQ.SCR.REGION)
            MOVE)
            (T (ACE.RETURN.CLOSEST.VERTEX MOVE (ACE.MT.SEQ.SCR.REGION))
```

(ACEEDITBM

(* MJD "23-May-86 15:51")

```
[LAMBDA (TYPE)
  (PROG (TEMPREGION TEMPBM NEWBM)
    (ACE.TELLEM "Select a Region inside the Sequence" T)
    (SETQ TEMPREGION (GETREGION NIL NIL 'ACE.EDIT.MOVE.REGION.ASST NIL))
    (SETQ TEMPBM (BITMAPCREATE (fetch (REGION WIDTH) of TEMPREGION)
      (fetch (REGION HEIGHT) of TEMPREGION)
      1))
    (BITBLT ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
      TEMPBM 0 0 NIL NIL 'INPUT 'REPLACE)
    (SETQ NEWBM (EDITBM TEMPBM))
    (BITBLT TEMPBM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
      NIL NIL 'INPUT 'ERASE)
    (BITBLT NEWBM 0 0 ABITMAP (ACE.MT.SCRX.SEQX (fetch (REGION LEFT) of TEMPREGION))
      (ACE.MT.SCRY.SEQY (fetch (REGION BOTTOM) of TEMPREGION))
      NIL NIL 'INPUT 'REPLACE))
```

(ACE.READBRUSHSHAPE

(* MJD "15-May-86 15:15")

```
[LAMBDA NIL
  (MENU (create MENU
    ITEMS _ ' (DIAGONAL VERTICAL HORIZONTAL SQUARE ROUND other]))
```

)

(* * TABLET AND SUPPORT FNS)

(DEFINEQ

(ACE.EDIT.POINT&CODE

[LAMBDA (DEVICE)

(* MD "19-Jun-85 16:56")

(* THIS MESS RETURNS (CODE X NIL) FOR THE DEVICE
SELECTED)

(* CODE IS NIL, VERTEX OR TOGGLE)

(* X AND Y ARE RELATIVE TO THE SEQUENCE. USUALLY THE SEQUENCE'S 0,0 IS AT THE WINDOW'S 0,0
THEREFORE, COORS IN THE A.C.W ARE SEQUENCE COORS, NOT WINDOW COORS.
E.S.L.DRAW TAKES CARE TO CORRECT FOR ANY DIFFERENCE)

```
(PROG (POINT&CODE XCOOR YCOOR)
  (ACE.GET.DEVICE.STATE DEVICE)
  (SETQ XCOOR (ACE.MT.SCRX.SEQX LASTMOUSEX))
  (SETQ YCOOR (ACE.MT.SCRY.SEQY LASTMOUSEY))
  [SETQ POINT&CODE (DECODEBUTTONS ' (LEFT MIDDLE RIGHT)
  [COND
    (POINT&CODE (FLIPCUSOR)
      (until (EQ LASTMOUSEBUTTONS 0) do (ACE.GET.DEVICE.STATE DEVICE))
      (FLIPCUSOR)
      (SETQ POINT&CODE (COND
        ((EQUAL POINT&CODE ' (LEFT))
         ' VERTEX)
        ((EQUAL POINT&CODE ' (MIDDLE))
         ' TOGGLE)
        ((EQUAL POINT&CODE ' (RIGHT))
         ' OPTIONS]
      )
    )
  (COND
    ((EQ DEVICE ' MOUSE)
     NIL)
    ((EQ DEVICE ' MM1201)
     (\SETCURSORPOSITION LASTMOUSEX LASTMOUSEY)))
  (ACE.UPD.CONTROL.WINDOW ' CURSOR (CONS XCOOR YCOOR))
  (RETURN (CONS POINT&CODE (CONS XCOOR YCOOR]))
```

(ACE.GET.DEVICE.STATE

[LAMBDA (DEVICE)

(* PmT "24-Apr-85 16:57")

(* Updates LASTMOUSEX LASTMOUSEY
LASTMOUSEBUTTONS LASTKEYBOARD based on DEVICE
(mouse or tablet (MM1201)))

```
(PROG (POINT&CODE)
  (COND
    ((EQ DEVICE ' MOUSE)
     (GETMOUSESTATE))
    ((EQ DEVICE ' MM1201)
     (SETQ POINT&CODE (ACE.MM1201POLL 1))
     (COND
       ((BITTEST (CDR POINT&CODE)
        64)
        NIL)
       (T (COND
          ((BITTEST (CDR POINT&CODE)
           2)
           (SETQ LASTMOUSEBUTTONS 1))
          ((BITTEST (CDR POINT&CODE)
           1)
           (SETQ LASTMOUSEBUTTONS 4))
          (T (SETQ LASTMOUSEBUTTONS 0)))
          (SETQ LASTMOUSEX (CAAR POINT&CODE))
          (SETQ LASTMOUSEY (CDAR POINT&CODE))
          (SETQ LASTKEYBOARD (\EVENTKEYS]))
```

(ACE.GET.DEVICE.STATE&CURSOR

[LAMBDA NIL

(* PmT "25-Apr-85 14:10")

(* Updates LASTMOUSEX LASTMOUSEY LASTMOUSEBUTTONS LASTKEYBOARD based on the window prop value of
DEVICE; Also, puts cursor info in status window (sequence referenced))

```
(PROG (DEVICE POINT&CODE)
  (SETQ DEVICE (WINDOWPROP ACE.CONTROL.WINDOW ' ACE.DEVICE))
  [COND
    ((EQ DEVICE ' MOUSE)
     (GETMOUSESTATE))
    ((EQ DEVICE ' MM1201)
     (SETQ POINT&CODE (ACE.MM1201POLL 1))
     (COND
       ((BITTEST (CDR POINT&CODE)
        64)
        NIL)
       (T (COND
          ((BITTEST (CDR POINT&CODE)
```

```
(PROG (XFACTOR DELTAY DELTAX XROUND YROUND BITMAPWIDTH BITMAPHEIGHT BITMAPBASE
      NEWBITMAP NEWHEIGHT-1 NEWBITMAPBASE NEWRASTERWIDTH ORIGBASE NEWBASE ORIGWORD NEWWORD XSTART
      YSTART ENDX ENDY ONLINE)
  (OR (type? BITMAP BITMAP)
    (\ILLEGAL.ARG BITMAP))
  (SETQ BITMAPWIDTH (fetch (BITMAP BITMAPWIDTH) of BITMAP))
  (SETQ BITMAPHEIGHT (fetch (BITMAP BITMAPHEIGHT) of BITMAP))
  (COND
    ((NUMBERP FACTOR)
      (SETQ XFACTOR FACTOR)
      (SETQ YFACTOR FACTOR))
    ((POSITIONP FACTOR)
      (SETQ XFACTOR (CAR FACTOR))
      (SETQ YFACTOR (CDR FACTOR)))
    (T (\ILLEGAL.ARG FACTOR)))
  [AND (FLOATP XFACTOR)
    (SETQ XFACTOR (FIX (FTIMES XFACTOR 100))
  [AND (FLOATP YFACTOR)
    (SETQ YFACTOR (FIX (FTIMES YFACTOR 100))
  (SETQ XFACTOR (IMIN SCREENWIDTH XFACTOR))
  (SETQ YFACTOR (IMIN SCREENHEIGHT YFACTOR))
  (COND
    ((ILESSP XFACTOR 101)
      (SETQ DELTAX 100)
      (SETQ XROUND (IQUOTIENT XFACTOR 2)))
    (T (SETQ DELTAX XFACTOR)
      (SETQ XROUND 50)))
  (COND
    ((ILESSP YFACTOR 101)
      (SETQ DELTAY 100)
      (SETQ YROUND (IQUOTIENT YFACTOR 2)))
    (T (SETQ DELTAY YFACTOR)
      (SETQ YROUND 50)))
  (SETQ NEWBITMAP (BITMAPCREATE (IQUOTIENT (IPLUS XROUND DELTAX (ITIMES (SUB1 BITMAPWIDTH)
                                                                    XFACTOR)))
                                100)
                                (IQUOTIENT (IPLUS YROUND DELTAY (ITIMES (SUB1 BITMAPHEIGHT)
                                                                    YFACTOR)))
                                100)
  1))
(* MAKE ALL VALUES QUICKLY AVAILABLE)
```

```

(SETQ HEIGHT-1 (SUB1 BITMAPHEIGHT))
(SETQ RASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of BITMAP))
(SETQ BITMAPBASE (fetch (BITMAP BITMAPBASE) of BITMAP)) (* AND THE NEW BITMAP VALUES)
(SETQ NEWHEIGHT-1 (SUB1 (fetch (BITMAP BITMAPHEIGHT) of NEWBITMAP)))
(SETQ NEWRASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of NEWBITMAP))
(SETQ NEWBITMAPBASE (fetch (BITMAP BITMAPBASE) of NEWBITMAP))
(* OK, CRANK IT OUT)
(* ORIGWORD AND NEWWORD ARE SORTA CACHED FOR
SPEED PURPOSES)
[for Y from 0 to HEIGHT-1
do [SETQ ORIGBASE (\ADDBASE BITMAPBASE (ITIMES RASTERWIDTH (IDIFFERENCE HEIGHT-1 Y)
(SETQ ONLINE NIL)
[for X from 0 to (SUB1 BITMAPWIDTH)
do [AND (ZEROP (IMOD X 16))
(SETQ ORIGWORD (\GETBASE ORIGBASE (LRSH X 4)
(* LOOK FOR STRINGS OF "ON" BITS;
THEN TREAT AS A LINE FOR TRANSLATIONAL PURPOSES)
(COND
[(BITTEST ORIGWORD (\WORDELT BITMASKARRAY (IMOD X 16)))
(OR ONLINE (AND (SETQ ONLINE T)
(SETQ XSTART X)
(SETQ YSTART Y]
)
(* JUST SKIP OVER BLANKS)
)
(T
(* SPELL THIS ALL OUT SO I CAN SEE WHAT'S GOIN' ON
HERE)
(SETQ XSTART (IQUOTIENT (IPLUS XROUND (ITIMES XSTART XFACTOR))
100))
(SETQ ENDY (IQUOTIENT (IPLUS (ITIMES YSTART YFACTOR)
YROUND DELTAY)
100))
(SETQ YSTART (IQUOTIENT (IPLUS YROUND (ITIMES YSTART YFACTOR))
100))
(SETQ ENDX (IQUOTIENT (IPLUS XROUND (ITIMES (SUB1 X)
XFACTOR))
100))
(for NY from YSTART to (SUB1 ENDY)
do (SETQ NEWWORD (\GETBASE [SETQ NEWBASE (\ADDBASE NEWBITMAPBASE
(ITIMES NEWRASTERWIDTH
(IDIFFERENCE NEWHEIGHT-1
NY]
(LRSH XSTART 4)))
(for NX from XSTART to ENDX do [AND (ZEROP (IMOD NX 16))
(SETQ NEWWORD (\GETBASE NEWBASE
(LRSH NX 4]
[SETQ NEWWORD (LOGOR NEWWORD
(\WORDELT BITMASKARRAY
(IMOD NX 16]
(AND (ZEROP (IMOD (ADD1 NX)
16))
(\PUTBASE NEWBASE (LRSH NX 4)
NEWWORD)))
(\PUTBASE NEWBASE (LRSH ENDX 4)
NEWWORD))
(SETQ ONLINE NIL]
(COND
(ONLINE
(* GOTTA CLEANUP AFTER THE LAST CASE)
(* THIS IN CASE WORKING ON A LINE THAT GOES TO END
(* GAWD! WHAT A WASTE O SPACE THIS IS.
FIX LATER)
(SETQ XSTART (IQUOTIENT (IPLUS XROUND (ITIMES XSTART XFACTOR))
100))
(SETQ ENDY (IQUOTIENT (IPLUS (ITIMES YSTART YFACTOR)
YROUND DELTAY)
100))
(SETQ YSTART (IQUOTIENT (IPLUS YROUND (ITIMES YSTART YFACTOR))
100))
(SETQ ENDX (IQUOTIENT (IPLUS XROUND (ITIMES (SUB1 BITMAPWIDTH)
XFACTOR))
100))
(for NY from YSTART to (SUB1 ENDY)
do (SETQ NEWWORD (\GETBASE [SETQ NEWBASE (\ADDBASE NEWBITMAPBASE
(ITIMES NEWRASTERWIDTH
(IDIFFERENCE NEWHEIGHT-1 NY]
(LRSH XSTART 4)))
(for NX from XSTART to ENDX do [AND (ZEROP (IMOD NX 16))
(SETQ NEWWORD (\GETBASE NEWBASE
(LRSH NX 4]
[SETQ NEWWORD (LOGOR NEWWORD
(\WORDELT BITMASKARRAY
(IMOD NX 16]
(AND (ZEROP (IMOD (ADD1 NX)
16))
(\PUTBASE NEWBASE (LRSH NX 4)
NEWWORD)))
(\PUTBASE NEWBASE (LRSH ENDX 4)
NEWWORD]

```


(RETURN NEWBITMAP))

(ACE.COMPILE.FRAME.ACE

[LAMBDA (PREC SUCC VERTICAL THRESHOLD)

(* MJD "28-May-86 17:14")

(* Calls the frame compiler on PREC and SUCC (bitmaps); returns the changes required to go from PREC to SUCC;
supplies defaults if not given. VERTICAL and THRESHOLD are special args to the compiler)

(* NEXT TWO ARE ARBITRARY DEFAULTS)

(OR VERTICAL (SETQ VERTICAL 16))

(OR THRESHOLD (SETQ THRESHOLD 50))

(ACE.UPD.CONTROL.WINDOW 'OPERATION "COMPILING")

(* ACE.COMPILE.FRAME PREC SUCC VERTICAL THRESHOLD)

(SETQ ACE.COMPIER (ADD.PROCESS `(ACE.COMPILE.FRAME %, PREC %, SUCC %, VERTICAL %, THRESHOLD))

(ACE.MM1201.INIT

[LAMBDA (INIT?)

(* PmT "24-Apr-85 17:16")

(* Inits the RS232 port and the MM1201 graphics tablet; If INIT? = ASK then the user is asked if s/he wants to init;
If = T then else auto init; else just the tablet is initialized)

(* DEFAULT RS232 IS 4800; DOESN'T QUITE WORK AT 9600)

(ACE.UPD.CONTROL.WINDOW 'OPERATION "INIT TABLET")

[COND

((OR (EQ INIT? T)

(AND (EQ INIT? 'ASK)

(ACE.CONFIRMIT "Click LEFT to Initialize Tablet." T)))

(RS232INIT (COND

((KEYDOWNP 'LSHIFT)

(OR (MENU (create MENU

ITEMS _ '(9600 4800 2400 1200 600 300 150 75)

TITLE _ "Select Baud Rate"

CENTERFLG _ T))

4800))

(T 4800))

8 NIL 1)

(forDuration 50 timerUnits 'MILLISECONDS)

(RS232CLEARBUFFER 'BOTH)

(RS232WRITEBYTE 32 T)

(forDuration 50 timerUnits 'MILLISECONDS]

(RS232CLEARBUFFER 'BOTH)

(* set tablet baud rate%:)

(* Set x and y scale factors%: USE SCREEN SIZE
(Scaling to window size is an interesting idea))

(RS232WRITEBYTE 114)

(RS232WRITEBYTE (LOGAND SCREENWIDTH 255))

(RS232WRITEBYTE (LRSH SCREENWIDTH 8))

(RS232WRITEBYTE (LOGAND SCREENHEIGHT 255))

(RS232WRITEBYTE (LRSH SCREENHEIGHT 8)

T)

(forDuration 50 timerUnits 'MILLISECONDS)

(RS232WRITEBYTE 68 T)

])

(* SET TABLET FOR POLLING MODE)

(ACE.MM1201.POLL

[LAMBDA (COUNT)

(* PmT "24-Apr-85 17:22")

(* Returns a point in the form ((X . Y) . CODE) Sends out the command to poll the pen;
receives data describing the pen's current state)

(* HACKED FROM MD "13-Jun-84 16:08")

(PROG (PT)

[AND (EQP COUNT 3)

(RETURN (ACE.MM1201.PROBLEM 'NODATA)

(RS232WRITEBYTE 80 T)

(* CONDUCT A POLL!)

[SETQ PT (LIST (RS232READBYTE 30 'MILLISECONDS)

(RS232READBYTE 15 'MILLISECONDS)

(RS232READBYTE 15 'MILLISECONDS)

(RS232READBYTE 15 'MILLISECONDS)

(RS232READBYTE 15 'MILLISECONDS]

(COND

((FMEMB NIL PT)

(RETURN (ACE.MM1201.POLL (ADD1 COUNT)))

(* If read screws up, try again (up to 3 times);
then tell user trouble)

)

(T (RETURN (CONS (CONS (LOGOR (CADR PT)

(LLSH (CADDR PT)

7))

(LOGOR (CADDR PT)

(LLSH (CAR (LAST PT))

7)))

(CAR PT))

(ACE.MM1201.PROBLEM

[LAMBDA (PROBLEM)

(* PmT "24-Apr-85 17:25")

(* Called if tablet ain't woikin right;
try a re-init)

(ACE.TELLEM "Tablet (MM1201) data problem." T)

```
(ACE.TELLEM "Will try to Re-Initialize Tablet" 'L)
(ACE.CONFIRMIT "Click ANY to continue." 'L 'ANY)
(ACE.MM1201.INIT 'ASK)
(ACE.MM1201.POLL 1))
```

(ACE.EDIT.CLEAR.ALL.MENUS

[LAMBDA NIL

(* PmT "24-Apr-85 17:27")
 (* THIS JUST ZAPS ALL MENUS AT LOAD TIME)
 (* MAKE THIS NICER SOME TIME)

```
(SETQ ACE.EDIT.FRAME.MENU NIL)
(SETQ ACE.EDIT.LINEART.ADJ.MENU NIL)
(SETQ ACE.EDIT.MOVE.MENU NIL)
(SETQ ACE.EDIT.TEXT.FONT.MENU NIL)
(SETQ ACE.EDIT.TEXT.SIZE.MENU NIL)
(SETQ ACE.EDIT.TEXT.FACE.MENU NIL)
(SETQ ACE.EDIT.TEXTURE.MENU NIL)
(SETQ ACE.EDIT.PUTDOWN.MENU NIL))
```

(ROTATEBM

[LAMBDA (SOURCE DEST ANGLE)

(* MJD "23-May-86 15:20")
 (* Original code by Kelly Roach (Roach.pa))


```
(PROG (SWIDTH SWIDTH2 SHEIGHT2 DWIDTH DWIDTH2 DHEIGHT2 DHEIGHT SIN COS AU BU CU AV BV CV U1 V1 U2 V2 DELTAU
      DELTAV)
  (CURSOR WAITINGCURSOR)
  [SETQ SWIDTH (COND
    ((WINDOWP SOURCE)
     (WINDOWPROP SOURCE 'WIDTH))
    ((BITMAPP SOURCE)
     (BITMAPWIDTH SOURCE))
    (T (\ILLEGAL.ARG SOURCE]
  (SETQ SWIDTH2 (IQUOTIENT SWIDTH 2))
  (SETQ SHEIGHT2 (IQUOTIENT (COND
    ((WINDOWP SOURCE)
     (WINDOWPROP SOURCE 'HEIGHT))
    ((BITMAPP SOURCE)
     (BITMAPHEIGHT SOURCE))
    (T (\ILLEGAL.ARG SOURCE)))
    2))
  [SETQ DWIDTH (COND
    ((WINDOWP DEST)
     (WINDOWPROP DEST 'WIDTH))
    ((BITMAPP DEST)
     (BITMAPWIDTH DEST))
    (T (\ILLEGAL.ARG DEST]
  (SETQ DWIDTH2 (IQUOTIENT DWIDTH 2))
  [SETQ DHEIGHT (COND
    ((WINDOWP DEST)
     (WINDOWPROP DEST 'HEIGHT))
    ((BITMAPP DEST)
     (BITMAPHEIGHT DEST))
    (T (\ILLEGAL.ARG DEST]
  (SETQ DHEIGHT2 (IQUOTIENT DHEIGHT 2))
  (SETQ SIN (SIN ANGLE))
  (SETQ COS (COS ANGLE))
  (SETQ AU COS)
  (SETQ BU SIN)
  (SETQ CU (FPLUS SWIDTH2 (FTIMES (FMINUS DWIDTH2)
    COS)
    (FTIMES (FMINUS DHEIGHT2)
    SIN)))
  (SETQ AV (FMINUS SIN))
  (SETQ BV COS)
  (SETQ CV (FPLUS SHEIGHT2 (FTIMES DWIDTH2 SIN)
    (FTIMES (FMINUS DHEIGHT2)
    COS)))
  (SETQ U1 CU)
  (SETQ V1 CV)
  (SETQ U2 (FPLUS (FTIMES AU DWIDTH)
    CU))
  (SETQ V2 (FPLUS (FTIMES AV DWIDTH)
    CV))
  (for Y from 0 to DHEIGHT do (SETQ DELTAU (FTIMES Y BU))
    (SETQ DELTAV (FTIMES Y BV))
    (TEXTURELINE SOURCE (FIXR (FPLUS U1 DELTAU))
      (FIXR (FPLUS V1 DELTAV))
      (FIXR (FPLUS U2 DELTAU))
      (FIXR (FPLUS V2 DELTAV))
      DEST 0 Y DWIDTH))
  (CURSOR T])
```

(RS232LOSTCHARFN[LAMBDA NIL
NIL])

(* PmT "20-Nov-84 14:19")

```
{MEDLEY}<lispusers>ACE>ACE.;1
```

Page 35

```
)  
  
(DECLARE%: EVAL@COMPILE  
  
(PUTPROPS ACE.POPPOS MACRO ((STACK X Y)  
                               (SETQ Y (pop STACK))  
                               (SETQ X (fetch (POSITION XCOORD) of Y))  
                               (SETQ Y (fetch (POSITION YCOORD) of Y))))  
  
(PUTPROPS ACE.PUSHPOS MACRO ((X Y STACK)  
                               (push STACK (CREATEPOSITION X Y))))  
  
(RPAQ ACE.EDIT.LINEART.SQUARE.CURSOR (CURSORCREATE '   
                                           'NIL 7 7))  
  
(RPAQ ACELOGOMAP
```



```
(RPAQ RS232LOSTCHARFN RS232LOSTCHARFN)
```

```
(* * MENU FOR ACE-EDIT)
```

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY  
  
(GLOBALVARS ACE.CONTROL.WINDOW ACE.EDIT.FRAME.MENU ACE.EDIT.LINEART.ADJ.MENU ACE.EDIT.MOVE.MENU  
             ACE.EDIT.TEXT.FONT.MENU ACE.EDIT.TEXT.SIZE.MENU ACE.EDIT.TEXT.FACE.MENU ACE.EDIT.TEXTURE.MENU  
             ACE.EDIT.PUTDOWN.MENU)  
  
(DECLARE%: DONTEVAL@LOAD DOCOPY  
  
(ACE.EDIT.CLEAR.ALL.MENUS)  
  
(PUTPROPS ACE COPYRIGHT ("Michel Denber" 1988 1993))
```

FUNCTION INDEX

ACE	2	ACE.EDIT.PUTDOWN.BITMAP	28	ACE.RETURN.CLOSEST.VERTEX	11
ACE.ANIMATE	3	ACE.EDIT.REDRAW.ABITMAP	31	ACE.RUN	3
ACE.ASKEM	10	ACE.EDIT.SETUP.EDIT.MENU	20	ACE.RUN.CURRENT.SEQ	4
ACE.ASKEM2	14	ACE.EDIT.TEXT	23	ACE.RUN.TRILLIUM	8
ACE.CHECKSTUFF	12	ACE.EDIT.TEXTURE.AREA	25	ACE.SCALE.BITMAP	31
ACE.COMPILE.FRAME	16	ACE.EDIT.TEXTURE.REGION	24	ACE.SCAN.BITMAPS	18
ACE.COMPILE.FRAME.ACE	33	ACE.EXTRACT	16	ACE.SCAN.PRIMBLOCKS	18
ACE.COMPUTE.AREA	18	ACE.EXTRACTBM	31	ACE.SCANLINESEEDFILL	27
ACE.CONFIRMIT	10	ACE.FETCH.BLOCK	18	ACE.SEQ.FETCH.HEIGHT	13
ACE.CREATE.CONTROL.MENU	13	ACE.FIGURE.OUT.WINDOW	10	ACE.SEQ.FETCH.WIDTH	13
ACE.CREATE.EDITING.BORDER	8	ACE.FILLWITHTEXTURE	27	ACE.SET.DEVICE	6
ACE.DECREMENT.FRAME	5	ACE.GET.A.FILE.NAME	9	ACE.SET.SEQ.CLIP.REGION	13
ACE.DEFINE.SEQ.WINDOW	10	ACE.GET.DEVICE.STATE	30	ACE.SETUP.CW.CLIPPING.REGIONS	11
ACE.DELAY	5	ACE.GET.DEVICE.STATE&CURSOR	30	ACE.TELLEM	10
ACE.DELAY.FRAME	5	ACE.GET.SEQ.FILE	5	ACE.TELLEM2	14
ACE.DELAY.FRAME.ASST	11	ACE.INCREMENT.FRAME	5	ACE.TRILLIUM	7
ACE.DELAY.SEQ	5	ACE.MAX.REGIONS	17	ACE.TRILLIUM.LOOP	7
ACE.DELETE.FRAME	6	ACE.MM1201.INIT	33	ACE.UPD.CLEAR.SET.LINE	13
ACE.EDIT	19	ACE.MM1201.PROBLEM	33	ACE.UPD.CONTROL.WINDOW	12
ACE.EDIT.CLEAR.ALL.MENUS	34	ACE.MM1201POLL	33	ACE.UPD.CONTROL.WINDOW2	14
ACE.EDIT.COMBINE.REGION	23	ACE.NEW.FRAME	4	ACE.UPD.CW.MULE	12
ACE.EDIT.CREATE.MENU.TEXTURES	28	ACE.NEW.SEQ.ASST	11	ACEEDITBM	29
ACE.EDIT.FRAME	19	ACE.NEW.SEQUENCE	4	ACEGETFRAME#	3
ACE.EDIT.LINEART	21	ACE.PICK.BEST.REGION	17	ACEGETREGIONFACTOR	20
ACE.EDIT.LINEART.ADJ	22	ACE.PUT.SEQ.FILE	9	ACEROTATEREGION	20
ACE.EDIT.LINEART.DRAW	22	ACE.QUICKDRAW&UPD	6	ACERUNLOOP	3
ACE.EDIT.LINEART.TRACKLINE	22	ACE.QUIT.ACE	4	ACESCALEREGION	21
ACE.EDIT.MOVE.REGION	23	ACE.QUIT.TRILLIUM	8	ACESETTHRESHOLD	16
ACE.EDIT.MOVE.REGION.ASST	29	ACE.READBRUSHSHAPE	29	ROTATEBM	34
ACE.EDIT.PAINT	25	ACE.RECONSTRUCT.FRAME	6	RS232LOSTCHARFN	34
ACE.EDIT.POINT&CODE	30	ACE.RESET.SEQ	4	SUBLIST	6

MACRO INDEX

ACE.MAC.CW.INFO.CLIP ...	14	ACE.MT.AWX.SCRX	1	ACE.MT.SCRY.AWY	1	ACE.MT.SEQY.AWY	2
ACE.MAC.CW.PROMPT.CLIP .	15	ACE.MT.AWX.SEQX	2	ACE.MT.SCRY.SEQY	1	ACE.MT.SEQY.SCRY	2
ACE.MAC.FETCH.HEIGHT ...	15	ACE.MT.AWY.SCRY	1	ACE.MT.SEQ.AW.REGION ...	1	ACE.POPPOS	35
ACE.MAC.FETCH.WIDTH ...	15	ACE.MT.AWY.SEQY	2	ACE.MT.SEQ.SCR.REGION ..	1	ACE.PUSHPOS	35
ACE.MAC.SEQ.CLIP	15	ACE.MT.SCRX.AWX	1	ACE.MT.SEQX.AWX	2		
ACE.MT.AW.SCR.POINT	1	ACE.MT.SCRX.SEQX	1	ACE.MT.SEQX.SCRX	2		

VARIABLE INDEX

ACE-EDITCOMS	19	ACE.BITMAP.MASK	16	ACE.PIXPERWORD	16
ACE-MAINCOMS	2	ACE.EDIT.LINEART.SQUARE.CURSOR ..	35	ACE.RIGHTMOUSE.CURSOR	15
ACE-PRIMCOMS	16	ACE.LEFTMOUSE.CURSOR	15	ACELOGOMAP	35
ACE.ALLMOUSE.CURSOR	15	ACE.MIDDLEMOUSE.CURSOR	15	RS232LOSTCHARFN	35

RECORD INDEX

ACE.BLIT	1	ACE.FRAME	1
----------------	---	-----------------	---
