```
  File created:   12-Jun-90 10:07:44  {DSK}<usr>local>lde>lispcore>library>MATMULT.;2

   changes to:    (VARS MATMULTCOMS)

 previous date:   22-Apr-87 09:55:51  {DSK}<usr>local>lde>lispcore>library>MATMULT.;1

   Read Table:    INTERLISP

      Package:    INTERLISP

       Format:    XCCS
```

```
(RPAQQ MATMULTCOMS
       (

;;; User entry points

        (DECLARE%: DONTCOPY DOEVAL@COMPILE (FILES FLOAT-ARRAY-SUPPORT))
        (FUNCTIONS %%MATMULT-N33 %%MATMULT-N44 DEGREES-TO-RADIANS IDENTITY-3-BY-3 IDENTITY-4-BY-4
               MAKE-HOMOGENEOUS-3-BY-3 MAKE-HOMOGENEOUS-3-VECTOR MAKE-HOMOGENEOUS-4-BY-4
               MAKE-HOMOGENEOUS-4-VECTOR MAKE-HOMOGENEOUS-N-BY-3 MAKE-HOMOGENEOUS-N-BY-4 MATMULT-133 MATMULT-144
               MATMULT-331 MATMULT-333 MATMULT-441 MATMULT-444 MATMULT-N33 MATMULT-N44 PERSPECTIVE-4-BY-4
               PROJECT-AND-FIX-3-VECTOR PROJECT-AND-FIX-4-VECTOR PROJECT-AND-FIX-N-BY-3 PROJECT-AND-FIX-N-BY-4
               ROTATE-3-BY-3 ROTATE-4-BY-4-ABOUT-X ROTATE-4-BY-4-ABOUT-Y ROTATE-4-BY-4-ABOUT-Z SCALE-3-BY-3
               SCALE-4-BY-4 TRANSLATE-3-BY-3 TRANSLATE-4-BY-4)


;;; Compiler options

        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY (LOCALVARS . T))
        (PROP FILETYPE MATMULT)))


;;; User entry points

(DECLARE%: DONTCOPY DOEVAL@COMPILE

(FILESLOAD FLOAT-ARRAY-SUPPORT)
)


(DEFMACRO %%MATMULT-N33 (N A-BASE B-BASE RESULT-BASE)
   `(CL:DO ((I 0 (CL:1+ I))
            (SOURCE-BASE ,A-BASE (\ADDBASE SOURCE-BASE 6))
            (DEST-BASE ,RESULT-BASE (\ADDBASE DEST-BASE 6))
            (MATRIX-BASE ,B-BASE))
           ((EQ I ,N))
        (%%MATMULT-133 SOURCE-BASE MATRIX-BASE DEST-BASE)))


(DEFMACRO %%MATMULT-N44 (N A-BASE B-BASE RESULT-BASE)
   `(CL:DO ((I 0 (CL:1+ I))
            (SOURCE-BASE ,A-BASE (\ADDBASE SOURCE-BASE 8))
            (DEST-BASE ,RESULT-BASE (\ADDBASE DEST-BASE 8))
            (MATRIX-BASE ,B-BASE))
           ((EQ I ,N))
        (%%MATMULT-144 SOURCE-BASE MATRIX-BASE DEST-BASE)))


(CL:DEFUN DEGREES-TO-RADIANS (DEGREES)
   (CL:* (FLOAT DEGREES)
         (CONSTANT (/ CL:PI 180.0))))


(CL:DEFUN IDENTITY-3-BY-3 (&OPTIONAL RESULT)
   (LET [(MATRIX (%%INSURE-ARRAY RESULT (3 3]
        (FILL-ARRAY MATRIX 0.0)
        (CL:DOTIMES (I 3)
            (ASET 1.0 MATRIX I I))
        MATRIX))


(CL:DEFUN IDENTITY-4-BY-4 (&OPTIONAL RESULT)
   (LET [(MATRIX (%%INSURE-ARRAY RESULT (4 4]
        (FILL-ARRAY MATRIX 0.0)
        (CL:DOTIMES (I 4)
            (ASET 1.0 MATRIX I I))
        MATRIX))


(CL:DEFUN MAKE-HOMOGENEOUS-3-BY-3 (&KEY A00 A01 A10 A11 A20 A21)
   (LET [(MATRIX (CL:MAKE-ARRAY '(3 3)
                        :ELEMENT-TYPE
                        'CL:SINGLE-FLOAT]
        (CL:IF A00
            (ASET (FLOAT A00)
                  MATRIX 0 0))
        (CL:IF A01
            (ASET (FLOAT A01)
                  MATRIX 0 1))
        (CL:IF A10
            (ASET (FLOAT A10)
                  MATRIX 1 0))
        (CL:IF A11
            (ASET (FLOAT A11)
                  MATRIX 1 1))
        (CL:IF A20
            (ASET (FLOAT A20)
                  MATRIX 2 0))
        (CL:IF A21
            (ASET (FLOAT A21)
                  MATRIX 2 1))
        (ASET 1.0 MATRIX 2 2)
        MATRIX))


(CL:DEFUN MAKE-HOMOGENEOUS-3-VECTOR (&OPTIONAL X Y)
   (LET [(V (MAKE-VECTOR 3 :ELEMENT-TYPE 'CL:SINGLE-FLOAT]
        (CL:IF X
            (ASET (FLOAT X)
                  V 0))
        (CL:IF Y
            (ASET (FLOAT Y)
                  V 1))
        (ASET 1.0 V 2)
        V))


(CL:DEFUN MAKE-HOMOGENEOUS-4-BY-4 (&KEY A00 A01 A02 A03 A10 A11 A12 A13 A20 A21 A22 A23 A30 A31 A32)
   (LET [(MATRIX (CL:MAKE-ARRAY '(4 4)
                        :ELEMENT-TYPE
                        'CL:SINGLE-FLOAT]
        (CL:IF A00
            (ASET (FLOAT A00)
                  MATRIX 0 0))
        (CL:IF A01
            (ASET (FLOAT A01)
                  MATRIX 0 1))
        (CL:IF A02
            (ASET (FLOAT A02)
                  MATRIX 0 2))
        (CL:IF A03
            (ASET (FLOAT A03)
                  MATRIX 0 3))
        (CL:IF A10
            (ASET (FLOAT A10)
                  MATRIX 1 0))
        (CL:IF A11
            (ASET (FLOAT A11)
                  MATRIX 1 1))
        (CL:IF A12
            (ASET (FLOAT A12)
                  MATRIX 1 2))
        (CL:IF A13
            (ASET (FLOAT A13)
                  MATRIX 1 3))
        (CL:IF A20
            (ASET (FLOAT A20)
                  MATRIX 2 0))
        (CL:IF A21
            (ASET (FLOAT A21)
                  MATRIX 2 1))
        (CL:IF A22
            (ASET (FLOAT A22)
                  MATRIX 2 2))
        (CL:IF A23
            (ASET (FLOAT A23)
                  MATRIX 2 3))
        (CL:IF A30
            (ASET (FLOAT A30)
                  MATRIX 3 0))
        (CL:IF A31
            (ASET (FLOAT A31)
                  MATRIX 3 1))
        (CL:IF A32
            (ASET (FLOAT A32)
                  MATRIX 3 2))
        (ASET 1.0 MATRIX 3 3)
        MATRIX))
```

```
(CL:DEFUN MAKE-HOMOGENEOUS-4-VECTOR (&OPTIONAL X Y Z)
   (LET [(V (MAKE-VECTOR 4 :ELEMENT-TYPE 'CL:SINGLE-FLOAT]
        (CL:IF X
            (ASET (FLOAT X)
                  V 0))
        (CL:IF Y
            (ASET (FLOAT Y)
                  V 1))
        (CL:IF Z
            (ASET (FLOAT Z)
                  V 2))
        (ASET 1.0 V 3)
        V))


(CL:DEFUN MAKE-HOMOGENEOUS-N-BY-3 (N &KEY INITIAL-ELEMENT)
   (LET [(MATRIX (CL:MAKE-ARRAY (LIST N 3)
                        :ELEMENT-TYPE
                        'CL:SINGLE-FLOAT]
        (CL:IF INITIAL-ELEMENT
            (FILL-ARRAY MATRIX (FLOAT INITIAL-ELEMENT)))
        (CL:DOTIMES (I N)
            (ASET 1.0 MATRIX I 2))
        MATRIX))


(CL:DEFUN MAKE-HOMOGENEOUS-N-BY-4 (N &KEY INITIAL-ELEMENT)
   (LET [(MATRIX (CL:MAKE-ARRAY (LIST N 4)
                        :ELEMENT-TYPE
                        'CL:SINGLE-FLOAT]
        (CL:IF INITIAL-ELEMENT
            (FILL-ARRAY MATRIX (FLOAT INITIAL-ELEMENT)))
        (CL:DOTIMES (I N)
            (ASET 1.0 MATRIX I 3))
        MATRIX))


(CL:DEFUN MATMULT-133 (VECTOR MATRIX &OPTIONAL RESULT)
   (%%TEST-ARRAY VECTOR (3))
   (%%TEST-ARRAY MATRIX (3 3))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (3)))
   (CL:IF (EQ VECTOR RESULT)
          (CL:ERROR "Results undefined if VECTOR reused"))
   (%%MATMULT-133 (%%GET-FLOAT-ARRAY-BASE VECTOR)
          (%%GET-FLOAT-ARRAY-BASE MATRIX)
          (%%GET-FLOAT-ARRAY-BASE RESULT))
   RESULT)


(CL:DEFUN MATMULT-144 (VECTOR MATRIX &OPTIONAL RESULT)
   (%%TEST-ARRAY VECTOR (4))
   (%%TEST-ARRAY MATRIX (4 4))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (4)))
   (CL:IF (EQ VECTOR RESULT)
          (CL:ERROR "Results undefined if VECTOR reused"))
   (%%MATMULT-144 (%%GET-FLOAT-ARRAY-BASE VECTOR)
          (%%GET-FLOAT-ARRAY-BASE MATRIX)
          (%%GET-FLOAT-ARRAY-BASE RESULT))
   RESULT)


(CL:DEFUN MATMULT-331 (MATRIX VECTOR &OPTIONAL RESULT)
   (%%TEST-ARRAY MATRIX (3 3))
   (%%TEST-ARRAY VECTOR (3))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (3)))
   (CL:IF (EQ MATRIX RESULT)
          (CL:ERROR "Results undefined if MATRIX reused"))
   (%%MATMULT-331 (%%GET-FLOAT-ARRAY-BASE MATRIX)
          (%%GET-FLOAT-ARRAY-BASE VECTOR)
          (%%GET-FLOAT-ARRAY-BASE RESULT))
   RESULT)


(CL:DEFUN MATMULT-333 (MATRIX-1 MATRIX-2 &OPTIONAL RESULT)
   (%%TEST-ARRAY MATRIX-1 (3 3))
   (%%TEST-ARRAY MATRIX-2 (3 3))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (3 3)))
   (CL:IF (EQ MATRIX-1 RESULT)
          (CL:ERROR "Results undefined if MATRIX-1 reused"))
   (%%MATMULT-333 (%%GET-FLOAT-ARRAY-BASE MATRIX-1)
          (%%GET-FLOAT-ARRAY-BASE MATRIX-2)
          (%%GET-FLOAT-ARRAY-BASE RESULT))
   RESULT)


(CL:DEFUN MATMULT-441 (MATRIX VECTOR &OPTIONAL RESULT)
   (%%TEST-ARRAY MATRIX (4 4))
   (%%TEST-ARRAY VECTOR (4))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (4)))
   (CL:IF (EQ MATRIX RESULT)
          (CL:ERROR "Results undefined if MATRIX reused"))
   (%%MATMULT-441 (%%GET-FLOAT-ARRAY-BASE MATRIX)
          (%%GET-FLOAT-ARRAY-BASE VECTOR)
          (%%GET-FLOAT-ARRAY-BASE RESULT))
   RESULT)


(CL:DEFUN MATMULT-444 (MATRIX-1 MATRIX-2 &OPTIONAL RESULT)
   (%%TEST-ARRAY MATRIX-1 (4 4))
   (%%TEST-ARRAY MATRIX-2 (4 4))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (4 4)))
   (CL:IF (EQ MATRIX-1 RESULT)
          (CL:ERROR "Results undefined if MATRIX-1 reused"))
   (%%MATMULT-444 (%%GET-FLOAT-ARRAY-BASE MATRIX-1)
          (%%GET-FLOAT-ARRAY-BASE MATRIX-2)
          (%%GET-FLOAT-ARRAY-BASE RESULT))
   RESULT)


(CL:DEFUN MATMULT-N33 (MATRIX-1 MATRIX-2 &OPTIONAL RESULT)
   (%%TEST-ARRAY MATRIX-1 (CL:* 3))
   (%%TEST-ARRAY MATRIX-2 (3 3))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (CL:* 3)
                       (CL:ARRAY-DIMENSIONS MATRIX-1)))
   (CL:IF (EQ MATRIX-1 RESULT)
          (CL:ERROR "Results undefined if MATRIX-1 reused"))
   (LET ((N (CL:ARRAY-DIMENSION MATRIX-1 0)))
        (CL:IF (NOT (EQ N (CL:ARRAY-DIMENSION RESULT 0)))
               (CL:ERROR "Dimensional mismatch"))
        (%%MATMULT-N33 N (%%GET-FLOAT-ARRAY-BASE MATRIX-1)
               (%%GET-FLOAT-ARRAY-BASE MATRIX-2)
               (%%GET-FLOAT-ARRAY-BASE RESULT))
        RESULT))


(CL:DEFUN MATMULT-N44 (MATRIX-1 MATRIX-2 &OPTIONAL RESULT)
   (%%TEST-ARRAY MATRIX-1 (CL:* 4))
   (%%TEST-ARRAY MATRIX-2 (4 4))
   (SETQ RESULT (%%INSURE-ARRAY RESULT (CL:* 4)
                       (CL:ARRAY-DIMENSIONS MATRIX-1)))
   (CL:IF (EQ MATRIX-1 RESULT)
          (CL:ERROR "Results undefined if MATRIX-1 reused"))
   (LET ((N (CL:ARRAY-DIMENSION MATRIX-1 0)))
        (CL:IF (NOT (EQ N (CL:ARRAY-DIMENSION RESULT 0)))
               (CL:ERROR "Dimensional mismatch"))
        (%%MATMULT-N44 N (%%GET-FLOAT-ARRAY-BASE MATRIX-1)
               (%%GET-FLOAT-ARRAY-BASE MATRIX-2)
               (%%GET-FLOAT-ARRAY-BASE RESULT))
        RESULT))


(CL:DEFUN PERSPECTIVE-4-BY-4 (PX PY PZ &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-4-BY-4 RESULT)))
        (ASET (FLOAT PX)
              MATRIX 0 3)
        (ASET (FLOAT PY)
              MATRIX 1 3)
        (ASET (FLOAT PZ)
              MATRIX 2 3)
        MATRIX))


(CL:DEFUN PROJECT-AND-FIX-3-VECTOR (3-VECTOR &OPTIONAL 2-VECTOR)
   (%%TEST-ARRAY 3-VECTOR (3))
   (COND
      [(NULL 2-VECTOR)
       (SETQ 2-VECTOR (CL:MAKE-ARRAY '(2]
      ([NOT (TYPEP 2-VECTOR '(CL:ARRAY CL:* (2]
       (CL:ERROR "Not a 2 vector: ~s" 2-VECTOR)))
   (LET ((3-VECTOR-BASE (%%GET-FLOAT-ARRAY-BASE 3-VECTOR)))
        (CL:DOTIMES (J 2)
            (ASET (UFIX (\GETBASEFLOATP 3-VECTOR-BASE (LLSH J 1)))
                  2-VECTOR J))
        2-VECTOR))


(CL:DEFUN PROJECT-AND-FIX-4-VECTOR (4-VECTOR &OPTIONAL 2-VECTOR)
   (%%TEST-ARRAY 4-VECTOR (4))
   (COND
      [(NULL 2-VECTOR)
       (SETQ 2-VECTOR (CL:MAKE-ARRAY '(2]
      ([NOT (TYPEP 2-VECTOR '(CL:ARRAY CL:* (2]
       (CL:ERROR "Not a 2 vector: ~s" 2-VECTOR)))
   (LET* ((4-VECTOR-BASE (%%GET-FLOAT-ARRAY-BASE 4-VECTOR))
          (DIVISOR (\GETBASEFLOATP 4-VECTOR-BASE 6)))
         (DECLARE (TYPE FLOATP DIVISOR))
         (CL:IF (UFEQP DIVISOR 1.0)
             (CL:DOTIMES (J 2)
                 (ASET (UFIX (\GETBASEFLOATP 4-VECTOR-BASE (LLSH J 1)))
                       2-VECTOR J))
             (CL:DOTIMES (J 2)
                 (ASET (UFIX (FQUOTIENT (\GETBASEFLOATP 4-VECTOR-BASE (LLSH J 1))
                                    DIVISOR))
                       2-VECTOR J)))
         2-VECTOR))


(CL:DEFUN PROJECT-AND-FIX-N-BY-3 (N-3-MATRIX &OPTIONAL N-2-MATRIX)
   (%%TEST-ARRAY N-3-MATRIX (CL:* 3))
   (COND
      [(NULL N-2-MATRIX)
       (SETQ N-2-MATRIX (CL:MAKE-ARRAY (LIST (CL:ARRAY-DIMENSION N-3-MATRIX 0)
                                             2]
      ([NOT (TYPEP N-2-MATRIX '(CL:ARRAY CL:* (CL:* 2]
       (CL:ERROR "Not an N by 2 array: ~s" N-2-MATRIX)))
   (LET ((N (CL:ARRAY-DIMENSION N-3-MATRIX 0)))
        (CL:IF (NOT (EQ N (CL:ARRAY-DIMENSION N-2-MATRIX 0)))
               (CL:ERROR "Dimensional mismatch"))
        (CL:DO ((I 0 (CL:1+ I))
                (N-3-BASE (%%GET-FLOAT-ARRAY-BASE N-3-MATRIX)
                       (\ADDBASE N-3-BASE 6)))
               ((EQ I N))
            (CL:DOTIMES (J 2)
                (ASET (UFIX (\GETBASEFLOATP N-3-BASE (LLSH J 1)))
                      N-2-MATRIX I J)))
        N-2-MATRIX))


(CL:DEFUN PROJECT-AND-FIX-N-BY-4 (N-4-MATRIX &OPTIONAL N-2-MATRIX)
   (%%TEST-ARRAY N-4-MATRIX (CL:* 4))
   (COND
      [(NULL N-2-MATRIX)
       (SETQ N-2-MATRIX (CL:MAKE-ARRAY (LIST (CL:ARRAY-DIMENSION N-4-MATRIX 0)
                                             2]
      ([NOT (TYPEP N-2-MATRIX '(CL:ARRAY CL:* (CL:* 2]
       (CL:ERROR "Not an N by 2 array: ~s" N-2-MATRIX)))
   (LET ((N (CL:ARRAY-DIMENSION N-4-MATRIX 0)))
        (CL:IF (NOT (EQ N (CL:ARRAY-DIMENSION N-2-MATRIX 0)))
               (CL:ERROR "Dimensional mismatch"))
        (CL:DO ((I 0 (CL:1+ I))
                (N-4-BASE (%%GET-FLOAT-ARRAY-BASE N-4-MATRIX)
                       (\ADDBASE N-4-BASE 8)))
               ((EQ I N))
            [LET ((DIVISOR (\GETBASEFLOATP N-4-BASE 6)))
                 (DECLARE (TYPE FLOATP DIVISOR))
                 (CL:IF (UFEQP DIVISOR 1.0)
                     (CL:DOTIMES (J 2)
                         (ASET (UFIX (\GETBASEFLOATP N-4-BASE (LLSH J 1)))
                               N-2-MATRIX I J))
                     (CL:DOTIMES (J 2)
                         (ASET (UFIX (FQUOTIENT (\GETBASEFLOATP N-4-BASE (LLSH J 1))
                                            DIVISOR))
                               N-2-MATRIX I J)))])
        N-2-MATRIX))


(CL:DEFUN ROTATE-3-BY-3 (RADIANS &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-3-BY-3 RESULT))
         (COSPHI (CL:COS RADIANS))
         (SINPHI (CL:SIN RADIANS)))
        (ASET COSPHI MATRIX 0 0)
        (ASET (- SINPHI)
              MATRIX 0 1)
        (ASET SINPHI MATRIX 1 0)
        (ASET COSPHI MATRIX 1 1)
        MATRIX))


(CL:DEFUN ROTATE-4-BY-4-ABOUT-X (RADIANS &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-4-BY-4 RESULT))
         (COSPHI (CL:COS RADIANS))
         (SINPHI (CL:SIN RADIANS)))
        (ASET COSPHI MATRIX 1 1)
        (ASET (- SINPHI)
              MATRIX 1 2)
        (ASET SINPHI MATRIX 2 1)
        (ASET COSPHI MATRIX 2 2)
        MATRIX))
```

```
(CL:DEFUN ROTATE-4-BY-4-ABOUT-Y (RADIANS &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-4-BY-4 RESULT))
         (COSPHI (CL:COS RADIANS))
         (SINPHI (CL:SIN RADIANS)))
        (ASET COSPHI MATRIX 0 0)
        (ASET (- SINPHI)
              MATRIX 2 0)
        (ASET SINPHI MATRIX 0 2)
        (ASET COSPHI MATRIX 2 2)
        MATRIX))

(CL:DEFUN ROTATE-4-BY-4-ABOUT-Z (RADIANS &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-4-BY-4 RESULT))
         (COSPHI (CL:COS RADIANS))
         (SINPHI (CL:SIN RADIANS)))
        (ASET COSPHI MATRIX 0 0)
        (ASET (- SINPHI)
              MATRIX 0 1)
        (ASET SINPHI MATRIX 1 0)
        (ASET COSPHI MATRIX 1 1)
        MATRIX))

(CL:DEFUN SCALE-3-BY-3 (SX SY &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-3-BY-3 RESULT)))
        (ASET (FLOAT SX)
              MATRIX 0 0)
        (ASET (FLOAT SY)
              MATRIX 1 1)
        MATRIX))

(CL:DEFUN SCALE-4-BY-4 (SX SY SZ &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-4-BY-4 RESULT)))
        (ASET (FLOAT SX)
              MATRIX 0 0)
        (ASET (FLOAT SY)
              MATRIX 1 1)
        (ASET (FLOAT SZ)
              MATRIX 2 2)
        MATRIX))

(CL:DEFUN TRANSLATE-3-BY-3 (TX TY &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-3-BY-3 RESULT)))
        (ASET (FLOAT TX)
              MATRIX 2 0)
        (ASET (FLOAT TY)
              MATRIX 2 1)
        MATRIX))

(CL:DEFUN TRANSLATE-4-BY-4 (TX TY TZ &OPTIONAL RESULT)
   (LET ((MATRIX (IDENTITY-4-BY-4 RESULT)))
        (ASET (FLOAT TX)
              MATRIX 3 0)
        (ASET (FLOAT TY)
              MATRIX 3 1)
        (ASET (FLOAT TZ)
              MATRIX 3 2)
        MATRIX))

;;; Compiler options

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)
)

(PUTPROPS MATMULT FILETYPE CL:COMPILE-FILE)

(PUTPROPS MATMULT COPYRIGHT ("Venue & Xerox Corporation" 1985 1986 1987 1990))
```

FUNCTION INDEX

MACRO INDEX

PROPERTY INDEX