Appendix B

EdenLisp Scripts

The following Scripts are written in the definitive notation EdenLisp.

Script 1. Tumbler Mixer Machine: Shaft Design
Calculations for the shaft size to carry the steady and impact loading on a tumbler mixer machine with a payload of four tonnes of powder of density about 1000 kg/m^3

```
; Tumbler Mixer Machine
; by AJ Cartwright
; August 1993

real : pi R1 R2  L2 L1  Ws Wp Wf Wm   ; all terms defined as used
real : F1 F2 K   Mm Ma Tm Ta
real : Oy Oe Os  D Do   RF Me Te Trt

Wf = 15000.0              ; Frame + flask weight (N)
Wp = 28000.0              ; Powder weight (N)
Ws = Wf + Wp              ; Static weight = 43000(N)
L1 = 2484.0               ; Bearing spacing (mm)

; Shock load
F1= sqrt(4*Wp*1200*K/9)   ; Assume 2/9 of powder shifts vertically by 1200mm
F = F1/2.0                ; F1 =  powder surface & flask vertical on fall = 223 109 N
K = 40000.0/12.0          ; F2 =  second case: flask horizontal = 111 554 N
                          ; K  =  stiffness of the shaft/flask system, estimated by a
                          ;  deflection of 12mm under 4 tonnes load  = 3333.3 N/mm
R1 = Wm/2.0               ; Bearing reaction R1 = 44316 N
R2 = -Wm/2.0              ; bearing reaction R2 = -44316 N

; Moments and torques
Mm = Wm * L1/10.0         ; BM = (Ws + mean shock) x L2 = 2.20162 e7 N.mm
Ma = F1 * L1/10.0         ; variable shock BM = 20 000 x L2 = 5.54203 e7 N.mm
Tm = 5 000 000            ; mean torque = 5,000 N.m
```

```
Ta = 250.0*F1                ; variable shock torque = 55 777 250 N.mm
                             ; mean torque/rev using bearing formula Wm = 21 453 Nm
                             ; Horizontal reaction on tie rod 0.5m from c/l = 42 906 N

; Min Shaft Diameter
Oy = 600.0                   ; yield strength of shaft material (MPa)
Oe = 400.0                   ; endurance strength (MPa)
RF = 2.0                     ; reserve factor
Te = Tm/Oy + Ta/Oe           ; effective torque (N mm)
Me = Mm/Oy + Ma/Oe           ; effective bending moment (N mm)
Trt= sqrt(Te^2+Me^2)         ; Equivalent rotating-bending load
d  =(32*RF*Trt/pi)^0.33      ; Minimum shaft diameter = 167.149 mm
De = 1.08*Do                 ; Hollow shaft with 25 mm wall thickness = 180.531mm
```

Script 2, 4-Bar Linkage

Design for 4-bar linkage with animation by varying the angle theta, between the driver, bar A, and the horizontal

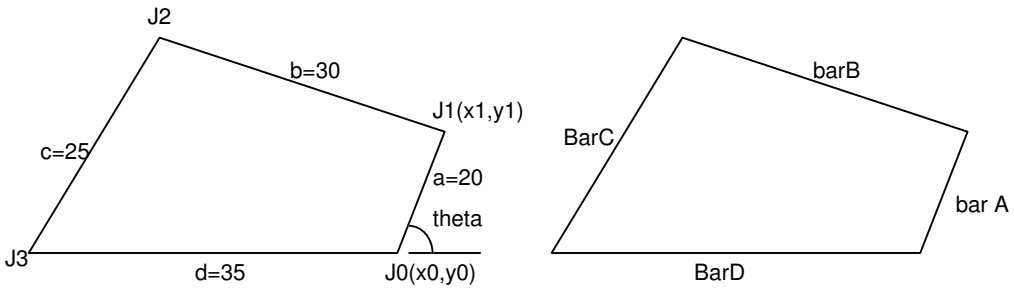

Fig B1.1 Notation and display for 4-bar linkage

```
; 4-bar linkage for EDENLISP
; by A.J.Cartwright
;
; Date: August 1991
; mods. 13 Dec 1992,
;       29 Mar 1993

 Lstr : nodes1                    ; list of strings for end labels of bars
LLstr : nodes2                    ; describes connections of  labels
 real : x y a b c d L M K S sq    ; declare variables (defined as used)
 real : theta delta
Lreal : J0 J1 J2 J3 J4 J5 J6 J7  ; points for ends of 4-bar,  label positions

nodes1= ["J0", "J1", "J2", "J3"] ;  labels for end points
nodes2= cedge(nodes1)            ;  see text below

theta = 1.0                      ; angle of driver (rad)
    a = 20.0                     ; length of driver arm (mm)
    b = 30.0                     ; length of cross member (mm)
```

```
    c = 25.0                              ; length of driven arm  (mm)
    d = 35.0                              ; distance between fixed pivots of a and c

    L = - sin(theta)                      ; computations to establish angle delta
    M = (d/a) + cos(theta)
    S = a*a + c*c + d*d - b*b
    K = d*cos(theta)/c + S/(2*a*c)
   sq = sqrt (L*L + M*M - K*K)
delta = 2*atan((L + sq)/(M - K))          ; compute angle delta from given data

    x = -(d + c*cos(delta2))              ;  hence find co-ords of point J2
    y = c*sin(delta2)

   J0 = [0.0,0.0]                         ; coords of fixed pivot of driver bar a
   J1 = [a*cos(theta), a*sin(theta)]      ;  coords of  moving end of driver bar a
   J2 = [x, y]                            ;  coords of  driven end of bar c
   J3 = [-d,0.0]                          ;  coords of  fixed pivot of driven bar c

frame : barb barf bard                    ; declare object variables

 barb = complex (nodes2)                  ; create variables making up the 4-bar
 barf = object(barb,[300,150],[5,5])      ; create 4-bar at (300,150) to scale*5
 bard = Wireframe (barf, "pline")         ; draw the 4-bar as a polyline

 Lpta = locate (j4,origin,size)
 Lptb = locate (j5,origin,size)
 Lptc = locate (j6,origin,size)
 Lptd = locate (j7,origin,size)
  Lht = 5.0
  Lwd = 0.85
 Lrot = 0.0

labela = label (Lpta, Lht, "bar a")       ; label bars by putting label at the
labelb = label (Lptb, Lht, "bar b")       ; mid points of each of the bars and
labelc = label (Lptc, Lht, "bar c")       ; the base
labeld = label (Lptd, Lht, "bar d")


 theta = 0.2                              ; now cycle through different values
 theta = 0.4                              ; theta to create the locus
 theta = 0.6
 theta = 0.8                              ; Later I found a better way to do this.
 theta = 1.0
 theta = 1.2
 theta = 1.4
 theta = 1.6
 theta = 1.8
 theta = 2.0
 theta = 2.2
 theta = 2.4
 theta = 2.6
```

Script 3. Parametric Drawing Frame

This script draws a drawing frame on any standard "A" size drawing sheets, orienting the sheets as landscape or portrait. A title-block is drawn in the bottom right hand corner with text inserted as per the labels in the definitions.

```
;; DRAWING FRAME  DRGFRAME.LSP
;; Drawing Frame for std drawings
;; by A.J.Cartwright
;; created 29 March 1993

LLreal : paper drg blk0 blk02 Dframe                    ; Declare variables
LLreal : blk1 blk2 blk3 blk4 blk5 blk6 blk7
LLreal : blk1f blk2f blk3f blk4f blk5f blk6f blk7f
frame  : blk1d blk2d blk3d blk4d blk5d blk6d blk7d
frame  : frameD
Lreal  : paprBL paprTL paprTR paprBR
Lreal  : drgBL drgTR drgTL drgBR
Lreal  : blkSC blkTL blkBL blkTR blkBR

str    : orient
int    : Asize BL TL x y
real   : len height blkDx blkDy

Asize  = 1                                              ; 'A1'-size paper, default
orient = "landscape"                                    ; orientation may be portriat
paper  = [paprBL,paprTL,paprTR,paprBR]                  ; coords of paper corners
drg    = [drgBL, drgTL, drgTR, drgBR]                   ; and drawing frame corners
blk0   = [blkBL, blkTL, blkTR]                          ; corner title block coords

len    = 1000*expt(2.0,0.25-Asize/2.0                   ;Standard 'A'-size paper
height = len/sqrt(2.0)                                  ; title block height
BL = 1                                                  ; Bottom left. These act as
TL = 2                                                  ; Top left      labels for projectn
x = 1
y = 2

paprBL = [0, 0]                                         ; paper bottom left coords
paprTR = if orient = "landscape" then [len, height]     ; paper top-right coords
                                 else [height, len]
drgBL  = [len/40.0,len/40.0]                            ; drg frame bottom left
drgTR  = vdiff(paprTR, drgBL)                           ;   top left coords
drgTL  = [projn(x,drgBL), projn(y,drgTR)])              ;   bottom right
drgBR  = [projn(x,drgTR), projn(y,drgBL)])              ;   bottom right
blkSc  = vscale([0-0.1,0.025],log(Asize+3.0))           ;-ve to move origin to B

blkBR  = paprBL                                         ; BR based on (0, 0)
blkTL  = locate([len,height], paprBL, blksc)            ;  and fraction
blkTR  = [projn(x,blkBR), projn(y,blkTL)]               ;  of paper size
blkBL  = [projn(x,blkTL), projn(y,blkBR)]
blk02  = vshear(blk0, [2.0,2.0])

blkDx  = projn(x,blkTR)-projn(x,blkTL)                  ; Dy and Dx
blkDy  = projn(y,blkTL)-projn(y,blkBL)                  ;  displacements

blk1   = vtrans(blk0, drgBR)                            ; locate coords of
blk2   = vtrans(blk1, [0.0, blkDy])                     ;  each frame for inserting text
blk3   = vtrans(blk2, [0.0, blkDy])
blk4   = vtrans(blk1, [(0.0-blkDx), 0.0])
blk5   = vtrans(blk4, [0.0, blkDy])
blk6   = vtrans(blk5, [0.0, blkDy])
```

```
blk7   = vtrans(blk02, projn(3,blk3))

Dframe = cedge(drg)                                  ; create frames themselves
blk1f  = pedge(blk1)
blk2f  = pedge(blk2)
blk3f  = pedge(blk3)
blk4f  = pedge(blk4)
blk5f  = pedge(blk5)
blk6f  = pedge(blk6)
blk7f  = pedge(blk7)

frameD = Wireframe(Dframe, "pline")                  ; draw the lines demarking
blk1D  = Wireframe(blk1f, "pline")                   ;  the title block
blk2D  = Wireframe(blk2f, "pline")
blk3D  = Wireframe(blk3f, "pline")
blk4D  = Wireframe(blk4f, "pline")
blk5D  = Wireframe(blk5f, "pline")
blk6D  = Wireframe(blk6f, "pline")
blk7D  = Wireframe(blk7f, "pline")

; TEXT LABELS
str   : labl1 labl2 labl3 labl4 labl5 labl6 labl7   ; declare label variables
str   : name date title namet datet titlet Univ
lreal : loc1 loc2 loc3 loc4 loc5 loc6 loc7
lreal : offset1 offset2
real  : txtht

name  = "A J Cartwright"                             ; label text
date  = "March 1993"
title  = "TEST"
namet  = "NAME"
datet  = "DATE"
titlet = "TITLE"
Univ   = "UNIVERSITY OF WARWICK"

txtht  = BlkDy*0.4                                   ; Text height
offset2= [blkDx*0.1, blkDy*0.25]  ; [1.0,2.5]       ; start position within frame
offset1= vshear(offset2,[2.0,2.0])

loc1   = vtrans(projn(BL, blk1),offset2)             ; Actual postions of text labels
loc2   = vtrans(projn(BL, blk2),offset2)
loc3   = vtrans(projn(BL, blk3),offset2)
loc4   = vtrans(projn(BL, blk4),offset2)
loc5   = vtrans(projn(BL, blk5),offset2)
loc6   = vtrans(projn(BL, blk6),offset2)
loc7   = vtrans(projn(BL, blk7),offset1)

labl1  = label(loc1, txtht, name)                    ; functions to place text on
labl2  = label(loc2, txtht, date)                    ;  the display itself
labl3  = label(loc3, txtht, title)
labl4  = label(loc4, txtht, namet)
labl5  = label(loc5, txtht, datet)
labl6  = label(loc6, txtht, titlet)
labl7  = label(loc7, txtht, univ)
```

Script 4. Elastic Hinge Design

The formulae for an elastic notch type hinge are given and then applied to the design of a monolith, a lever system for amplifying the input displacement by a factor of about 50.

; Monolith.Lsp

```
; by A.J.Cartwright
; New version  5 Apr 93
; revision    20 Apr 93

real  : Th R Z L E Omax
real  : F Mmax K Kt Qmax
real  : thetam lamda

E = 205000.0                                  ; Young' s Modulus steel (MPa)
z = 10.0                                      ; hinge width (mm)
Th = 0.2                                      ; hinge thickness at smallest, mm
R = 2.0                                       ; radius of hinge, mm
L = 25.0                                      ; length of lever arm, mm
Omax = 250.0                                  ; max strength of material, MPa
F = 400.0                                     ; force on lever: N

K    = 0.166 + (0.565 * th / R)               ; stress concentration factor
Mmax = Z*th*th*Omax / (6*Kt)                  ; max bending moment
Kt   = 0.325 + ((2.7*th)+(5.4*R))/(8*R + th)) ; total geometry factor
thetam = 4*K*R*Omax / (Kt*E*th)               ; max angle of displacement, rad
lamda  = E*Z*th*th*th / (6*K*R*L*L)           ; stiffness of system N/mm
Qmax   = thetam*L                             ; max displacement of lever mm
```

;Topology of the basic monolith hinge is just a cuboid.
;Its shape is a particular realisation of the basic cuboid

```
Lreal : NL FL FR NR
real  : ENL EFL EFR ENR Extr                  ; block vertices
real  : OO                                    ; origin
real  : BX BY BZ
real  : HX HY HZ LX LY LZ

BX = 50.0 * R                                 ; block dimensions
BY = 20.0 * R                                 ; as a function of R
BZ = evalID(Z)

HX = 2*R+Th                                   ; hinge dimensions
HY = 2*R
HZ = evalID(Z)

LX = evalID(HX)                               ; lever dimensions
LY = evalID(L)
LZ = evalID(Z)

OO  = [0,0,0]                                 ; origin

NL  = [0,0,0]                                 ; Base :Near/Left
FL  = [0,1,0]                                 ; Far/Left
FR  = [1,1,0]                                 ; Far/Right
NR  = [1,0,0]                                 ; Near/Right

Extr= [0,0,1]                                 ; extrusion vector
```

```
                                                    ; Top : Extruded/Near/Left etc
EFL = vtrans(FL,Extr)
EFR = vtrans(FR,Extr)
ENR = vtrans(NR,Extr)

LLstr : cuboidV
frame : cuboid
frame : block1  Vlever  Hlever  Hhinge  Vhinge
frame : block1D VleverD HleverD HhingeD VhingeD
str   : blocksh horizH vertH

cuboidV = extrude (["NL", "FL", "FR", "NR"])
cuboid  = complex (cuboidV)

block1 = object (cuboid, OO, [BX,BY,BZ])        ; base block
Vlever = object (cuboid, OO, [LX,LY,LZ])        ; dummy vertical lever
Hlever = object (cuboid, OO, [LY,LX,LZ])        ; dummy horizontal lever
Vhinge = object (cuboid, OO, [HX,HY,HZ])        ; dummy horizontal hinge
Hhinge = object (cuboid, OO, [HY,HX,HZ])        ; dummy vertical hinge

blocksh = "POLYHEDRON"
HorizH  = "Hhinge"
VertH   = "Vhinge"

block1D = Wireframe (block1, blocksh)
HleverD = Wireframe (Hlever, blocksh)
VleverD = Wireframe (Vlever, blocksh)
HhingeD = Wireframe (Hhinge, horizh)
VhingeD = Wireframe (Vhinge, verth)

frame : Hhinge1 Hhinge1D                        ; Put in particular levers and hinges
frame : Hhinge2 Hhinge2D
frame : Hhinge3 Hhinge3D
frame : Hhinge4 Hhinge4D

frame : Hlever1 Hlever1D
frame : Hlever2 Hlever2D
frame : Vlever1 Vlever1D
Lreal : O1 O2 O3 O4 I

 I = [1,1,1]
O1 = [13.0,13.0,0]
O2 = Vtrans(O1,[HY,0,0])
O3 = Vtrans(O2,[L,0,0])
O4 = Vtrans(O3,[HY,0,0])

Hhinge1 = object (Hhinge, O1, I)                ;lever1
Hlever1 = object (Hlever, O2, I)
Hhinge2 = object (Hhinge, O3, I)

Hhinge3 = object (Hhinge1, [0,20,0], I)         ;lever2
Hlever2 = object (Hlever1, [0,20,0], I)
Hhinge4 = object (Hhinge2, [0,20,0], I)

Vlever1 = object (cuboid, O4, [HX,20.0+HX,LZ])
                                                ; Realise (display) objects in
                                                ; appropriate surface shapes
Block1D  = Surface (block1, blocksh)            ; outer block
```

```
Hhinge1D = Surface (Hhinge1, HorizH)
Hlever1D = Surface (Hlever1, blocksh)
Hhinge2D = Surface (Hhinge2, HorizH)

Hhinge3D = Surface (Hhinge3, HorizH)
Hlever2D = Surface (Hlever2, blocksh)
Hhinge4D = Surface (Hhinge4, HorizH)

Vlever1D = Surface (Vlever1, blocksh)
```

Script 5. Graph Plotting

The function at the commencement of the listing is used to create the array of coordinates. It illustrates the complexity of functions that underlie EdenLisp. Such functions need to be designed to suit particular applications. Once done the definitions are easy to manipulate.

```
; GRAPH.LSP
; by AJ Cartwright
; October 1993
(defun mkgraf2 (Npts xrange yrange xyfn$
                    / mkgrh xylst x y fxy)
    (defun mkintL (N)
     (if (>= N 0)

     (append (mkintL (1- N)) (list N))
    ))

    (defun mkgrh (xylst func)
     (cond
      ((null xylst) nil)
      (T
       (setq x (caar xylst))
       (cons (list x y (eval (read func)))
             (mkgrh (cdr xylst) func)
    ))))

 (setq xylst
  (mapcar
  '(lambda (x)
    (list (* (/ (float xrange) Npts) x)
          (* (/ (float yrange) Npts) x)
   ))
   (mkintL Npts)
 ))
 (setq fxy (catlex (statmt (lex xyfn$))))
 (mapcar
  '(lambda (lst)
    (setq y (cadr lst))
    (mkgrh xylst fxy)
   ) xylst
))
```

; Associate function values
; with list members
; This is a separate help function
; It makes a list of int (1 2 3 4 5 ...)
; by recursion through 0.. N

; This is a separate help function
; It creates a list
; of y values calculated from
; inserting x-values into
; the analytical function "func"
; and returning (x,y) coords list
; It is recursive through xylist
; end of help function

; Function starts here

; replace integers by (x,y) coords
; = a set of x values in x-range
; = a set of y values in y-range
; by using an initial set
; of integers obtained here

; hand over actual (x,y) coords
; to EdenLisp
; having once computed the y value
; by passing the x values to the
; help function from the
; list created by the setq xylst above

; EdenLisp Script

```
str    : func
int    : Npts
```

; declare variables

```
real   : xrange yrange
lreal  : origin
frame  : gpts gline flist

Npts   = 50                                          ; Number of points on one graph
xrange = 40.0                                        ; range of x coords
yrange = 40.0                                        ; range of y coords
func   = "cos(sqrt(x^2 + y^2))"                      ; func = graph of analytic curve

flist  = mkgraf (Npts, xrange, yrange, func)         ; compute set of z coords
origin = [100.0,100.0,100.0]                         ; new origin
gpts   = object(flist, origin, [1,1,1])              ; move (x,y,z) set to new origin
gline  = wireframe (gpts, "carpet")                  ; and display as a carpet graph
```

Script 6 Denture Design

Script 6*a*. 2D mouth profile and tooth form

The first part of this script generates the list of points that go to make up each of the archetypal tooth shapes. Actual dummy points are created in order to make it easy to edit. Actions are used to generate all the teeth required.

```
;  DENTAL2.LSP
;  by AJ Cartwright
;  11th November 1993
;  rev 1st Dec 1993

; Dental features
; UpperLeft, UpperRight, LowerLeft, LowerRight
6 or 8 e.g.  UL1 UR7 LL4 LR8
;   Tooth   condition:   N.natural,   M.missing,
A.artificial
; Tooth changes allowed  N->M, M->A only
; Types: incisor canine molar wisdom
; Artificial features: Saddle, pad=compressive
support, hook=tensile support

; 2D Topology of tooth
; C of G must be within the region and on an
arc
; consists of a closed polyedge of N points

; TOPOLOGY
str : Mkincisor Mkcanine Mkmolar Mkwisdom mkpad
llreal : incisor canine molar wisdom pad

                                                     ; Actions  A_lst
Mkincisor = A_lst ("incisor","i","lreal",10)         ; generates labels for
Mkcanine  = A_lst ("canine", "c","lreal",10)         ; tooth shape points
Mkmolar   = A_lst ("molar",  "m","lreal",16)         ; creating lists such as
Mkwisdom  = A_lst ("wisdom", "w","lreal",16)         ; (w1,w2,w3....w12)
Mkpad     = A_lst ("pad", "p", "lreal",12)

;;;  GEOMETRY
i1 = [ 17,  0]                       ;    incisor tooth shape
i2 = [ 15,  9]
i3 = [  7, 12]                       ;      4       3
```

```
i4 = [ -5, 13]
i5 = [-14,  7]
i6 = [-17,  1]
i7 = [-13, -7]
i8 = [ -7,-11]
i9 = [  2,-13]
i10= [ 13,-11]

c1 = [ 15,  0]
c2 = [ 17, 12]
c3 = [  5, 13]
c4 = [ -5, 13]
c5 = [-17, 12]
c6 = [-15,  0]
c7 = [-10, -8]
c8 = [ -6,-13]
c9 = [  6,-13]
c10= [ 10, -8]

m1 =  [ 20,  0]
m2 =  [ 19,  7]
m3 =  [ 15, 11]
m4 =  [ 10, 15]
m5 =  [  0, 14]
m6 =  [-10, 15]
m7 =  [-15, 11]
m8 =  [-19,  7]
m9 =  [-20,  0]
m10 = [-19, -7]
m11 = [-15,-11]
m12 = [-10,-15]
m13 = [  0,-14]
m14 = [ 10,-15]
m15 = [ 15,-11]
m16 = [ 19, -7]

w1 =  [ 21,  0]
w2 =  [ 20,  8]
w3 =  [ 16, 12]
w4 =  [ 11, 16]
w5 =  [  0, 15]
w6 =  [-11, 16]
w7 =  [-16, 12]
w8 =  [-20,  8]
w9 =  [-21,  0]
w10 = [-20, -8]
w11 = [-16,-12]
w12 = [-11,-16]
w13 = [  0,-15]
w14 = [ 11,-16]
w15 = [ 16,-12]
w16 = [ 20, -8]


p1 =  [ 5.0, 0.0]
p2 =  [ 4.0, 3.0]
p3 =  [ 1.0, 4.0]
p4 =  [ 0.0, 5.0]
p5 =  [-1.0, 4.0]
```

```
; 5                2
;6         +          1
; 7               10
;    8           9
```

```
;       canine tooth shape

;          4      3
;5                      2
; 6          +         1
;    7              10
;         8       9
```

```
;          molar shape

;   6         5          4
; 7                    3
; 8                     2
; 9          +           1
; 10                    16
;   11                      15
;      12         13        14
```

```
;          wisdom tooth shape

;    6           5          4
; 7                       3
; 8                        2
; 9             +           1
; 10                       16
;   11                           15
;      12          13          14
```

```
;       artificial pad shape
```

```
p6 =  [-4.0, 3.0]
p7 =  [-5.0, 0.0]
p8 =  [-4.0,-3.0]
p9 =  [-1.0,-4.0]
p10 = [ 0.0,-5.0]
p11 = [ 1.0,-4.0]
p12 = [ 4.0,-3.0]
```

; Dental arrangement of mouth

```
real   : pi rt Rx Ry ang Z ScM                    ; declare variables used below
lreal  : scaleA
real    : ang1 ang2 ang3 ang4 ang5 ang6 ang7      ; scale used in display
ang8
real   : mouth male female child Nteeth           ; type of tooth

male   = 1.0                                      ; Scale for  mouth size
female = 0.9
child  = 0.7
Nteeth = if ScM=0.7 then 6.0 else 8.0
pi     = 3.14159265
rt     = pi/2.0

mouth  = male                                     ; define current mouth shape
ScM    = evalid(mouth)                            ; use its real value
Rx     = 200.0*ScM                                ; elliptical shape of mouth minor axis
Ry     = 250.0*ScM                                ; elliptical shape of mouth major axis
Z      = 1.0                                      ; Z scale (if 3D added)
SCALEa = if ScM=0.7 then [0.9,0.9,1.0]            ; scale Rxy
                    else [ScM,ScM,1.0]

Nteeth = if ScM=0.7 then 6.0 else 8.0             ; child has no wisdom teeth
ang    = pi/Nteeth*2.24                           ; position increment round ellipse

ang1   = (Nteeth + 0.2)*ang                       ; actual position of first tooth
ang2   = (Nteeth - 0.8)*ang                       ;  and 2nd tooth.. etc
ang3   = (Nteeth - 1.8)*ang
ang4   = (Nteeth - 2.8)*ang
ang5   = (Nteeth - 3.9)*ang
ang6   = (Nteeth - 5.0)*ang
ang7   = (Nteeth - 6.1)*ang
ang8   = (Nteeth - 7.2)*ang

str       :  tooth  toothp  natural  artificial  ; declare strings
missing
natural = "cpline"                                ; tooth is shown as a polyline
artificial = "fill"                               ; artificial tooth is shown hatched
tooth   = natural                                 ; define current tooth type
toothp  = if ScM=0.7 then "" else tooth


str  : URtd URpd URid URp URt URi URd
Lstr : UR UR1 UR2 UR3 UR4 UR5 UR6 UR7 UR8

URtd= "llreal: ?1?2typ "                          ; Dummy text used with actions
URpd= "lreal : ?1?2pos "                          ;    to create the definitions
URid= "frame : ?1?2ins ?1?2dsp"                   ;    for each tooth shape.
URp     =     "?1?2pos     =     [Rx*cos(ang?2),  ; Middle of tooth profile
Ry*sin(ang?2)]"
URt = "?1?2typ = rotobj(?3, rt-ang?2,  Z)"        ; orientation & type of tooth
URi   =  "?1?2ins   =   object(?1?2typ,   ?1?2pos,  ; instance of tooth
scalea)"
```

```
URd = "?1?2dsp =  wireframe(?1?2ins,?4)"          ; display the profile

UR  = [URtd, URpd, URid, URp, URt, URi, URd]

UR1=A_repl(["UR","1","incisor","tooth"], UR)
UR2=A_repl(["UR","2","incisor","tooth"], UR)
UR3=A_repl(["UR","3","canine","tooth"], UR)
UR4=A_repl(["UR","4","molar","tooth"], UR)
UR5=A_repl(["UR","5","molar","tooth"], UR)
UR6=A_repl(["UR","6","molar","tooth"], UR)
UR7=A_repl(["UR","7","wisdom","toothp"], UR)
UR8=A_repl(["UR","8","wisdom","toothp"], UR)

str  : ULp ULt
Lstr : UL UL1 UL2 UL3 UL4 UL5 UL6 UL7 UL8

ULp = "?1?2pos = Vshear(UR?2pos, [-1,1])"
ULt = "?1?2typ = rotobj(?3, ang?2-rt,  Z)"
UL  = [URtd, URpd, URid, ULp, ULt, URi, URd]

UL1=A_repl(["UL","1","incisor","tooth"], UL)
UL2=A_repl(["UL","2","incisor","tooth"], UL)
UL3=A_repl(["UL","3","canine","tooth"], UL)
UL4=A_repl(["UL","4","molar","tooth"], UL)
UL5=A_repl(["UL","5","molar","tooth"], UL)
UL6=A_repl(["UL","6","molar","tooth"], UL)
UL7=A_repl(["UL","7","wisdom","toothp"], UL)
UL8=A_repl(["UL","8","wisdom","toothp"], UL)

str  : LLp LLi
Lstr : LL LL1 LL2 LL3 LL4 LL5 LL6 LL7 LL8

LLp="?1?2pos = Vshear(UR?2pos, [1,-1])"
LLi="?1?2ins    =    object(UL?2typ,    ?1?2pos,
scaleA)"

LL = [URtd, URpd, URid, LLp, LLi, URd]

LL1=A_repl(["LL","1","incisor","tooth"], LL)
LL2=A_repl(["LL","2","incisor","tooth"], LL)
LL3=A_repl(["LL","3","canine","tooth"], LL)
LL4=A_repl(["LL","4","molar","tooth"], LL)
LL5=A_repl(["LL","5","molar","tooth"], LL)
LL6=A_repl(["LL","6","molar","tooth"], LL)
LL7=A_repl(["LL","7","wisdom","toothp"], LL)
LL8=A_repl(["LL","8","wisdom","toothp"], LL)

str  : LRp LRi
Lstr : LR LR1 LR2 LR3 LR4 LR5 LR6 LR7 LR8

LRp="?1?2pos = Vshear(UR?2pos, [-1,-1])"
LRi="?1?2ins    =    object(UR?2typ,    ?1?2pos,
scaleA)"

LR = [URtd, URpd, URid, LRp, LRi, URd]
LR1=A_repl(["LR","1","incisor","tooth"], LR)
LR2=A_repl(["LR","2","incisor","tooth"], LR)
LR3=A_repl(["LR","3","canine","tooth"], LR)
LR4=A_repl(["LR","4","molar","tooth"], LR)
LR5=A_repl(["LR","5","molar","tooth"], LR)
```

```
LR6=A_repl(["LR","6","molar","tooth"], LR)
LR7=A_repl(["LR","7","wisdom","toothp"], LR)
LR8=A_repl(["LR","8","wisdom","toothp"], LR)

llreal : PADtyp
frame : PADins PADdsp
padtyp = rotobj(pad, rt-ang1,  Z)
padins = object(PADtyp, UR1pos, scalea)
paddsp = wireframe(PADins,artificial)
```

Script 6*b* Denture Plate

The following script for the generation of the dental plate needs to be run after the first script as it makes use of the definitions there.

```
;  DENTAL3.LSP
;  Plate design
;  by AJ Cartwright
;  1st Dec 1993

str   : t1 mkplat
lstr  : mkp12 mkp45 mkp23 mkp67
llreal : plate
frame : plateO plateD
lreal : UR23pl UR45pl UL23pl UL45pl

t1 = "?1?2?3pl = midpt(?1?2pos, ?1?3pos)"
mkp12  = A_repl (["UR","2","3"], [t1])      ; Position of gaps
mkp45  = A_repl (["UR","4","5"], [t1])      ; over which denture
mkp23  = A_repl (["UL","2","3"], [t1])      ; is to act
mkp67  = A_repl (["UL","4","5"], [t1])
mkplat = A_lst ("plate","pl","lreal",28)    ; create list of 28 labels

pl1=projn(7,UR3ins)                         ; profile of denture
pl2=projn(6,UR3ins)                         ; takes values from
pl3=projn(5,UR3ins)                         ; the appropriate
pl4=projn(4,UR3ins)                         ; tooth forms
pl5=projn(3,UR3ins)
pl6=projn(2,UR3ins)

pl7=projn(6,UR4ins)                         ; and joins the teeth
pl8=projn(5,UR4ins)                         ; together
pl9=projn(4,UR4ins)
pl10=projn(3,UR4ins)
pl11=projn(2,UR4ins)
pl12=projn(1,UR4ins)
pl13=projn(16,UR4ins)
pl14=projn(15,UR4ins)

pl15=projn(11,UL4ins)                       ; crosses over to the
pl16=projn(10,UL4ins)                       ; opposite quarter of
pl17=projn(9,UL4ins)                        ; the mouth
pl18=projn(8,UL4ins)
pl19=projn(7,UL4ins)
pl20=projn(6,UL4ins)
pl21=projn(5,UL4ins)
```

```
pl22=projn(4,UL4ins)

pl23=projn(5,UL3ins)
pl24=projn(4,UL3ins)
pl25=projn(3,UL3ins)
pl26=projn(2,UL3ins)
pl27=projn(1,UL3ins)
pl28=projn(10,UL3ins)

plateO = object(plate, [0,0,0], scaleA)      ; make the object
plateD = wireframe(plateO, natural)          ; display as polyline

; TEXT LABELS
real   : txtht
lreal  : offset ofset2

txtht  = 10.0
offset = [0, 20.0]
ofset2 = [-55.0,5]

str    : labl1 labl2 labl3 labl4
lreal  : loc1 loc2 loc3 loc4
loc1   = vtrans(projn(1, UR3ins),offset)     ; position the labels
loc2   = vtrans(projn(1, UR4ins),offset)
loc3   = vtrans(projn(1, UL3ins),ofset2)
loc4   = vtrans(projn(1, UL4ins),ofset2)

labl1  = label(loc1, txtht, "UR3")           ; label tooth "UR3"
labl2  = label(loc2, txtht, "UR4")
labl3  = label(loc3, txtht, "UL3")
labl4  = label(loc4, txtht, "UL4")
```