Solid Vertical Path Models

Twisted tower section as a solid

If you are using AutoCAD version 2007 or greater, a solid model version of this form can be developed by using the same computed points as the surface version required for each floor section.

This example uses function PROG04, the twist version of the tower, any of the other variations can be modified in the same fashion to generate a solid model.

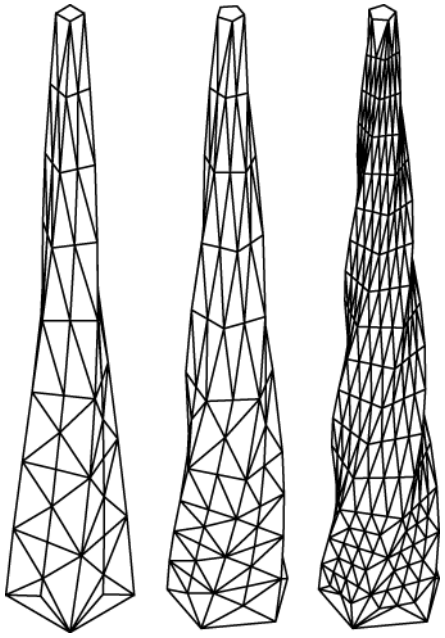

Figure 5.77a: PROG04a, converting floor sections to a solid volume, smooth fit

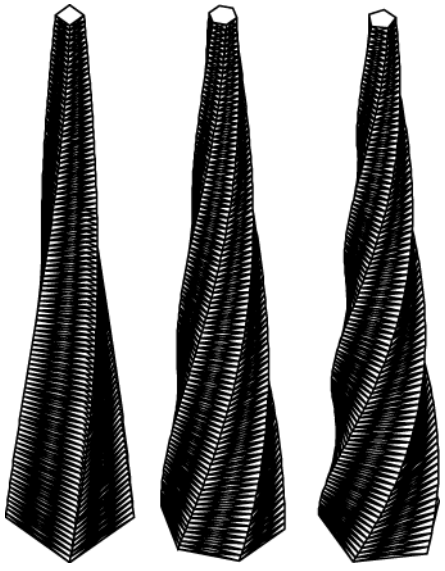

Figure 5.77b: PROG04a, converting floor sections to a solid volume, ruled

These examples include four sides at 90 degree rotation, six sides at 180 degrees, and five sides at 270 degrees.

Copy PROG04 to PROG04a. Remove the code sections for adding the top and bottom closing points and remove the section counter.

Change the 3DMESH command from:

```
; start mesh
 (command ".3DMESH"
   (+ (+ NumLevels 1) 2)
   (+ NumSides 1))
```

To starting a new selection list:

```
; start LOFT selection list
(setq plist (ssadd))
```

Add the start of the polyline before the polygon section is computed:

```
; start polyline
(command ".PLINE")
```

Change the adding of the 3DMESH points from:

```
; add to mesh
(command npnt)
```

To:

```
; add to polyline
(command npnt)
```

After the repeat for each floor polygon is completed, close the polygon and add it to the selection list:

```
; close polyline
(command "c")
; add to LOFT list
(setq plist
 (ssadd (entlast) plist))
```

At the completion of the floor loop, execute the LOFT command on the sections in the selection list, add:

```
 ; loft sections
 (command ".ZOOM" "e")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set LOFTNORMALS =1 for smooth
 ; =0 for ruled
 (setvar "LOFTNORMALS" 1)
 ; set LOFTPARAM to default
 (setvar "LOFTPARAM" 7)
 ; loft sections
 (command ".LOFT" plist "" "")
 ; turn on history
 (setvar "SOLIDHIST" 1)
```

A complete description of the values of these system variable can be found in the previous section.

The LOFTPARAM variable is set to its default value of 7.

Set the LOFTNORMALS variable to either smooth fit or ruled.

If other settings are required, check AutoCAD Help for a complete description of all the system variables that pertain to the LOFT command.

Sample script file for PROG04a:

```
;-----------------------------------
(prog04a)
; Layer name:
LAYER04a_3
; number of levels
100
```

```
; level height
10'
; number polygon sides
5
; radius start
90'
; radius end
20'
; rotation
270
;-----------------------------------

Completed function PROG04a:

;-----------------------------------
(defun prog04a ()
 (graphscr)
 ; tower by polygon repeats
 ; twist
 (setvar "CMDECHO" 0)
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get curve parameters
 (princ "\nProg04a - Tower")
 (setq TowerLayer
   (getstring
   "\nEnter layer name: "))
 (setq NumLevels
   (getint
   "\nEnter number of levels: "))
 (setq LevelHeight
   (getdist
   "\nEnter level height: "))
 (setq NumSides
   (getint
   "\nEnter number of polygon
   sides: "))
 (setq RadStart
   (getdist
   "\nEnter start radius: "))
 (setq RadEnd
   (getdist
   "\nEnter end radius: "))
 (setq RotTotal
   (getreal
   "\nEnter total rotation
   angle: "))
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; set elev
 (setq LevelElev 0.0)
 ; start LOFT selection list
 (setq plist (ssadd))
 ; radius inc
 (setq RadInc
   (/ (- RadEnd RadStart)
   NumLevels))
 (setq Rad RadStart)
 ; rotation inc
 (setq RotInc
   (/ RotTotal NumLevels))
 (setq Rotang 0.0)
 ; create each level
 (repeat (+ NumLevels 1)
  ; draw polygon
  (setq panginc (/ 360.0 NumSides))
  (setq pang (/ panginc 2.0))
  ; start polyline
  (command ".PLINE")
  (repeat (+ NumSides 1)
   ; compute point
   (setq xpt (+ (nth 0 cpnt)
     (* Rad (sin (dtr pang)))))
   (setq ypt (+ (nth 1 cpnt)
     (* Rad (cos (dtr pang)))))
   ; rotate point
   (setq rxpt
     (- (* xpt (cos (dtr Rotang)))
     (* ypt (sin (dtr Rotang)))))
   (setq rypt
     (+ (* xpt (sin (dtr Rotang)))
     (* ypt (cos (dtr Rotang)))))
   (setq npnt
     (list rxpt rypt LevelElev))
   ; add to polyline
   (command npnt)
   ; inc ang
   (setq pang (+ pang panginc))
  )
  ; close polyline
  (command "c")
  ; add to LOFT list
  (setq plist
   (ssadd (entlast) plist))
  ; inc radius
  (setq Rad (+ Rad RadInc))
  ; inc elev
  (setq LevelElev
    (+ LevelElev LevelHeight))
  ; inc rotation
  (setq Rotang (+ Rotang Rotinc))
 )
 (command ".ZOOM" "e")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set LOFTNORMALS =1 for smooth
 ; =0 for ruled
 (setvar "LOFTNORMALS" 1)
 ; set LOFTPARAM to default
 (setvar "LOFTPARAM" 7)
 ; loft sections
 (command ".LOFT" plist "" "")
 ; turn on history
 (setvar "SOLIDHIST" 1)
 (setvar "CMDECHO" 1)
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 (princ)
)
;-----------------------------------
```

Combining floor sections for more complex tower configurations

Many complex floor plan configurations can be created by a combination of methods described in the previous section. In most cases a floor plan can be created by computing the edge points. A floor plan can also be created by combining individual shapes.

Some other interesting possibilities are able to be explored if the entire tower is a solid which can be combined with other solids to form more complex configurations. In these cases each individual floor plan can be extracted by using the SECTION command on the combined solid at each floor height.

For example, in the function series PROG16, an elliptical floor plan, is used to create a polar array of similar forms, then UNIONed, and then the SECTION command passed on the combined solid to extract all the individual floor plans including accumulating the total gross floor area.

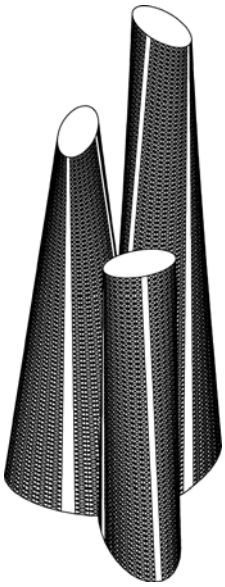

Figure 5.78a: PROG16a1, three elliptical towers combined, isometric view

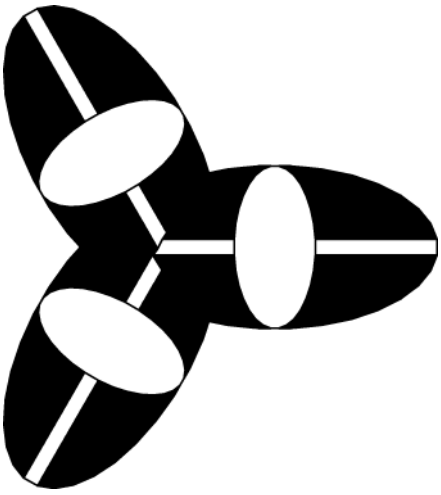

Figure 5.78b: PROG16a1, three elliptical towers combined, plan view

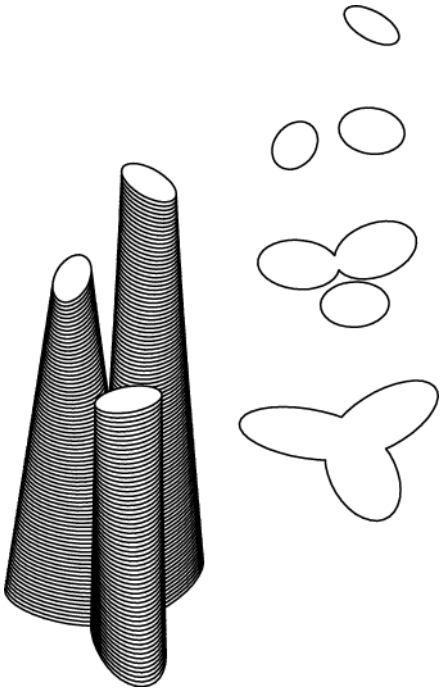

Figure 5.78c: PROG16a1, three elliptical towers combined, individual floor plans

Area dn perimetere results from the example of the three elliptical towers combined are:

Area=3170274 Perm=85913

The function series PROG16 includes the primary function PROG16a1 and secondary functions PROG16b, and PROG16c.

Function PROG16b is a copy of PROG06 converted to a callable function to generate a single elliptical tower. Function PROG16c can take the current solid and cut individual sections from it. Function PROG16a1 is the primary function which calls PROG16b three times, once for each tower, and then calls PROG16c to generate floor plans from the combined towers.

The outline for PROG16a1 is as follows:

```
;-----------------------------------
(defun prog16a1 ()
 (graphscr)
 ; tower by polygon repeats
 ; elliptical form
 ; set initial values for layer and
 ; system setup
  .
 ; generate first tower with (prog16b)
  .
 ; generate second tower with (prog16b)
 ; and rotate it
  .
 ; generate third tower with (prog16b)
 ; and rotate it
  .
 ; combine towers
  .
 ; do floor sections with (prog16c)
  .
 ; display area and completed tower
  .
(princ)
)
;-----------------------------------
```

Completed function PROG16a1:

```
;-----------------------------------
(defun prog16a1 ()
 (graphscr)
 ; tower by polygon repeats
 ; elliptical form
 (setvar "CMDECHO" 0)
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get tower parameters
 (princ "\nProg16a1 - Tower")
 (setq TowerLayer "prog16a1")
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; turn off undo
 (command ".UNDO" "end")
 ; first tower
 (setq NumLevels 100)
 (setq LevelHeight (* 12 10.0))
 (setq NumSides 32)
 (setq XRadStart (* 12 120.0))
 (setq XRadEnd (* 12 30.0))
 (setq YRadStart (* 12 60.0))
 (setq YRadEnd (* 12 60.0))
 (setq adist (* 12 -80.0))
 (prog16b)
 ; second tower
 (setq NumLevels 80)
 (prog16b)
 ; rotate
 (setq apnt (list
  (+ (nth 0 cpnt) adist)
  (nth 1 cpnt) (nth 2 cpnt)))
 (setq rang 120)
 (command ".ROTATE" "last" ""
  apnt rang)
 ; third tower
 (setq NumLevels 60)
 (prog16b)
 ; rotate
 (setq apnt (list
  (+ (nth 0 cpnt) adist)
  (nth 1 cpnt) (nth 2 cpnt)))
 (setq rang 240)
 (command ".ROTATE" "last" ""
  apnt rang)
 ; combine towers
 (command ".UNION" "all" "")
 (setq obj1 (ssadd (entlast)))
 (command ".VIEW" "top")
 (command ".ZOOM" "e")
 ; do floor sections
 (setq tdata (prog16c))
 (setq tarea (nth 0 tdata))
 (setq tperm (nth 1 tdata))
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (princ "\n") (princ "Area=")
 (princ (rtos (/ tarea 144) 2 0))
 (princ "  ") (princ "Perm=")
 (princ (rtos (/ tperm 12) 2 0))
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 ; turn on history
 (setvar "SOLIDHIST" 0)
 ; turn on undo
 (command ".UNDO" "begin")
 (setvar "CMDECHO" 1)
 (princ)
)
;-----------------------------------
```

Function PROG16b is based on a copy of PROG06 with the get commands removed. This function requires PROG16a1 to define the following variables: NumLevels, LevelHeight, NumSides, XRadStart, XRadEnd, YRadStart, YRadEnd, and adist. Review function PROG06 on the meaning of these variables.

Function PROG16b creates a LOFTed solid from the parameters set.

Completed function PROG16b:

```
;-----------------------------------
(defun prog16b ( / plist XRadInc XRad
 YRadInc YRad panginc pang xpt ypt
 npnt LevelElev)
 ; tower by polygon repeats
 ; elliptical form
 ; set elev
 (setq LevelElev 0.0)
 ; start LOFT selection list
 (setq plist (ssadd))
 ; radius inc
 (setq XRadInc
   (/ (- XRadEnd XRadStart)
   NumLevels))
 (setq XRad XRadStart)
 ; radius inc
 (setq YRadInc
   (/ (- YRadEnd YRadStart)
   NumLevels))
 (setq YRad YRadStart)
 ; create each level
 (repeat (+ NumLevels 1)
  ; draw polygon
  (setq panginc (/ 360.0 NumSides))
  (setq pang (/ panginc 2.0))
  ; start polyline
  (command ".PLINE")
  (repeat (+ NumSides 1)
   ; compute point
   (setq xpt (+ (nth 0 cpnt)
     (* XRad (sin (dtr pang)))))
   (setq ypt (+ (nth 1 cpnt)
     (* YRad (cos (dtr pang)))))
   (setq npnt
     (list xpt ypt LevelElev))
   ; add to polyline
   (command npnt)
   ; inc ang
   (setq pang (+ pang panginc))
  )
  ; close polyline
  (command "c")
  (command ".ZOOM" "e")
  ; add to LOFT list
  (setq plist
   (ssadd (entlast) plist))
  ; inc radius
  (setq XRad (+ XRad XRadInc))
  (setq YRad (+ YRad YRadInc))
  ; inc elev
  (setq LevelElev
    (+ LevelElev LevelHeight))
 )
 (command ".ZOOM" "e")
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set LOFTNORMALS =1 for smooth
 ; =0 for ruled
 (setvar "LOFTNORMALS" 0)
 ; set LOFTPARAM to default
 (setvar "LOFTPARAM" 7)
 ; loft sections
 (command ".LOFT" plist "" "")
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 (princ)
)
;-----------------------------------
```

Function PROG16c requires PROG16a1 to define the variables: LevelHeight, cpnt, and obj1; floor-to-floor height, location of tower, and selection list of the tower from which to create the sections.

Review function PROG16c, how the maximum and minimum dimensions of the tower are computed using the EXTMAX and EXTMIN system variable. These dimensions are required to locate the individual sections away from the tower itself; and how the SECTION command generates a REGION which is relocated next to the tower and also used to compute the floor area and perimeter.

Completed function PROG16c:

```
;-----------------------------------
(defun prog16c ( / xmax ymax xmax
 xmin ymin zmin xdim zdim tarea tperm
 zheight spnt mpnt nlevels)
 ; section through solid
 ; get exts
 (setq xmax
  (nth 0 (getvar "EXTMAX")))
 (setq ymax
  (nth 1 (getvar "EXTMAX")))
 (setq zmax
  (nth 2 (getvar "EXTMAX")))
 (setq xmin
  (nth 0 (getvar "EXTMIN")))
 (setq ymin
  (nth 1 (getvar "EXTMIN")))
 (setq zmin
  (nth 2 (getvar "EXTMIN")))
 ; compute xdim
 (setq xdim (- xmax xmin))
 ; compute zmax
 (setq zdim (- zmax 0.2))
 (setq nlevels
  (fix (/ zdim LevelHeight)))
 ; total area and perm
 (setq tarea 0.0)
 (setq tperm 0.0)
 ; start
 (setq zheight (+ 0.0 0.1))
 ; get sections
 (repeat nlevels
  ; section cut pt
  (setq spnt (list
   (nth 0 cpnt) (nth 1 cpnt)
   zheight))
  (command ".SECTION" obj1 ""
   "xy" spnt)
  ; section move pt
  (setq mpnt (list
   (+ (nth 0 spnt) (* xdim 1.5))
   (nth 1 spnt) (nth 2 spnt)))
  (command ".MOVE" "last" ""
   spnt mpnt)
  ; area
  (command ".AREA" "o" "last")
  (setq tarea
   (+ tarea (getvar "AREA")))
  (setq tperm
   (+ tperm (getvar "PERIMETER")))
  (command ".ZOOM" "e")
  ; inc z
  (setq zheight
   (+ zheight LevelHeight))
  )
; return area/perimeter
(list tarea tperm)
)
;-----------------------------------
```

Function PROG16a1 can be extended to any combination of LOFTed solid towers that are UNIONed into a single solid by replacing the function PROG16b, so that PROG16c can generate individual floor plans.

Laser cut tower model

In function PROG16c individual sections were cut from the solid model. The same concept can be used to create the layout sections for laser cutting.

Function PROG16a2 creates a single elliptical tower and thencalls function RPOG16d to create the sections for laser cutting.

The variable that control the laser cut sections include:

```
; cutting bed size or
; max sheet size
(setq blen 24)
(setq bwid 18)
; material thickness
(setq bthick (/ 1.0 8.0))
; registration size
(setq regsize (/ 3.0 16.0))
; model height
(setq mheight 10)
; model id
(setq modelid "A")
```

The laser cut sections are drawn next to the tower. Color assignments are used for cuts, red, and etched text, green. The tower is scaled to the specified height. Each section includes a registration marker and text that identifies the model and section number. A layout boundary is also included to represent the sheet size being cut. This can be used to estimate the amount of material need fort eh laser cuts.

Review each of these functions and the methods used to create the laser cut sections.

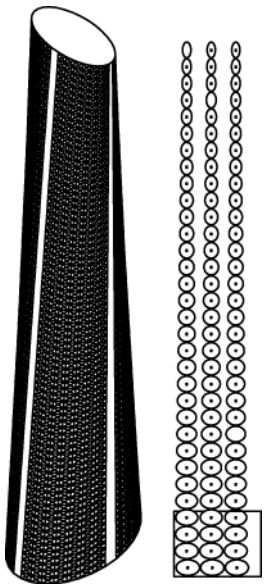

Figure 5.79a: PROG16a2, laser cut drawing from elliptical solid

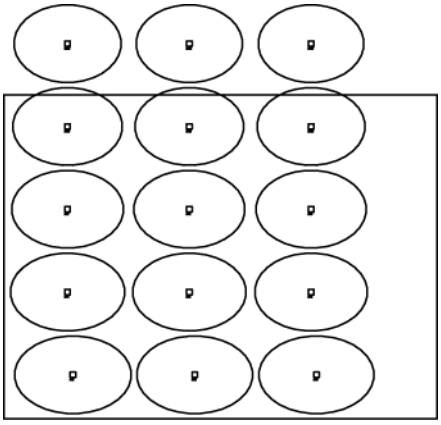

Figure 5.79b: PROG16a2, detail of each laser cut floor

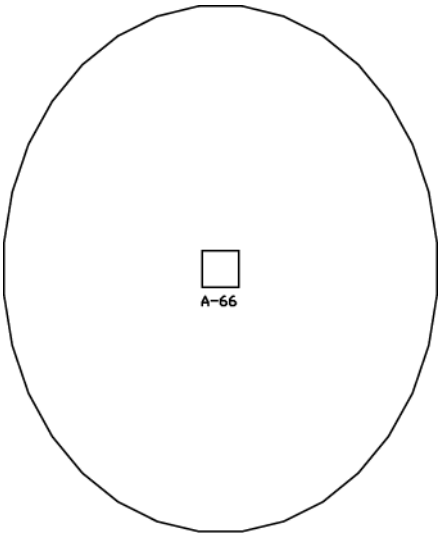

Figure 5.79c: PROG16a2, detail of the registration annotation on each laser cut

Completed function PROG16a2:

```
;-----------------------------------
(defun prog16a2 ()
 (graphscr)
 ; tower by polygon repeats
 ; elliptical form
 (setvar "CMDECHO" 0)
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get tower parameters
 (princ "\nProg16a2 - Tower")
 (setq TowerLayer "prog16a2")
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; set color
 (setvar "CECOLOR" "BYLAYER")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; turn off undo
 (command ".UNDO" "end")
 ; first tower
 (setq NumLevels 60)
 (setq LevelHeight (* 12 10.0))
 (setq NumSides 32)
 (setq XRadStart (* 12 120.0))
 (setq XRadEnd (* 12 30.0))
 (setq YRadStart (* 12 60.0))
 (setq YRadEnd (* 12 60.0))
 (prog16b)
 (setq obj1 (ssadd (entlast)))
 (command ".VIEW" "top")
 (command ".ZOOM" "e")
 ; cutting bed size or
 ; max sheet size
 (setq blen 24)
 (setq bwid 18)
 ; material thickness
 (setq bthick (/ 1.0 8.0))
 ; registration size
 (setq regsize (/ 3.0 16.0))
 ; model height
 (setq mheight 10)
 ; model id
 (setq modelid "A")
 ; do laser cuts
 (setq tdata (prog16d))
 (setq tarea (nth 0 tdata))
 (setq tperm (nth 1 tdata))
 (princ "\n") (princ "Area=")
 (princ (rtos (/ tarea 144) 2 0))
 (princ "  ") (princ "Perm=")
 (princ (rtos (/ tperm 12) 2 0))
 ; turn on history
 (setvar "SOLIDHIST" 0)
 ; turn on undo
 (command ".UNDO" "begin")
 ; reset color
 (setvar "CECOLOR" "BYLAYER")
 (setvar "CMDECHO" 1)
 (princ)
)
;-----------------------------------
```

Completed function PROG16d:

```
;-----------------------------------
(defun prog16d ( / xmax ymax xmax
 xmin ymin zmin xdim ydim zdim
 tarea tperm zcut spnt pnt1 pnt2
 zheight spnt nlevels nsect ncuts
 xsloc ysloc xslocinc yslocinc
 xloc yloc)
 ; laser layout through solid
 ; get exts
 (setq xmax
  (nth 0 (getvar "EXTMAX")))
 (setq ymax
  (nth 1 (getvar "EXTMAX")))
 (setq zmax
  (nth 2 (getvar "EXTMAX")))
 (setq xmin
  (nth 0 (getvar "EXTMIN")))
 (setq ymin
  (nth 1 (getvar "EXTMIN")))
 (setq zmin
  (nth 2 (getvar "EXTMIN")))
 ; compute xdim
 (setq xdim (- xmax xmin))
 (setq ydim (- ymax ymin))
 (setq zdim (- zmax zmin))
 ; compute scale
 (setq mscale (/ mheight zdim))
 ; compute cuts and height
 (setq ncuts
  (fix (/ (* zdim mscale)
  bthick)))
 (setq zcut
  (/ (- zdim 0.2) ncuts))
 ; layout locs
 (setq xsloc xdim)
 (setq ysloc 0.0)
 (setq xslocinc
  (+ (* xdim mscale) bthick))
 (setq yslocinc
  (+ (* ydim mscale) bthick))
```

```
(setq xloc xsloc)
(setq yloc ysloc)
; total area and perm
(setq tarea 0.0)
(setq tperm 0.0)
; start
(setq zheight (+ 0.0 0.1))
(setq nsect 1)
; get sections
(repeat ncuts
 ; set cut color
 (setvar "CECOLOR" "1")
 ; section cut pt
 (setq spnt (list
  (nth 0 cpnt) (nth 1 cpnt)
  zheight))
 (command ".SECTION" obj1 ""
  "xy" spnt)
 ; area
 (command ".AREA" "o" "last")
 (setq tarea
  (+ tarea (getvar "AREA")))
 (setq tperm
  (+ tperm (getvar "PERIMETER")))
 (command ".ZOOM" "e")
 (command ".SCALE"
  "last" "" spnt mscale)
 ; section move pt
 (setq pnt1 (list
  (- (nth 0 cpnt)
  (* (/ xdim 2) mscale))
  (- (nth 1 cpnt)
  (* (/ ydim 2) mscale))
  zheight))
 (setq pnt2 (list xloc yloc 0.0))
 (command ".MOVE" "last" ""
  pnt1 pnt2)
 ; registration
 (setq pnt1 (list
  (+ xloc (* (/ xdim 2) mscale))
  (+ yloc (* (/ ydim 2) mscale))))
 (command ".RECTANGLE"
  (list
  (- (nth 0 pnt1) (/ regsize 2))
  (- (nth 1 pnt1) (/ regsize 2)))
  (list
  (+ (nth 0 pnt1) (/ regsize 2))
  (+ (nth 1 pnt1) (/ regsize 2))))
 ; set text color
 (setvar "CECOLOR" "3")
 ; cut id
 (command ".TEXT"
  (list (- (nth 0 pnt1) (/ 1.0 16.0))
  (- (nth 1 pnt1) regsize))
  (/ 1.0 16.0) "0"
  (strcat modelid "-" (itoa nsect)))
 ; check for next cut
 (setq xloc (+ xloc xslocinc))
 (if (>
  (+ xloc xslocinc) (+ xsloc blen))
   (progn
    (setq xloc xsloc)
    (setq yloc (+ yloc yslocinc))
 ))
 ; inc z cut
 (setq zheight
  (+ zheight zcut))
 ; inc section
 (setq nsect (+ nsect 1))
)
; set layout color
(setvar "CECOLOR" "7")
; cutting bed
(setq pnt1 (list
 (- xsloc (* bthick 2))
 (- ysloc (* bthick 2))))
(setq pnt2 (list
 (+ (nth 0 pnt1) blen)
 (+ (nth 1 pnt1) bwid)))
(command ".RECTANGLE" pnt1 pnt2)
; layout
(command ".ZOOM" "w" pnt1 pnt2)
; return area/perimeter
(list tarea tperm)
)
;-----------------------------------
```

With some minor modifications functions PROG16c and PROG16d could be used as standalone functions for section cutting and laser cutting layouts for any solid. Function PROG16a1 demonstrated a single tower duplicated three times. Another variation of the combined towers is to generate the tower form but reverse the rotation on the second tower.

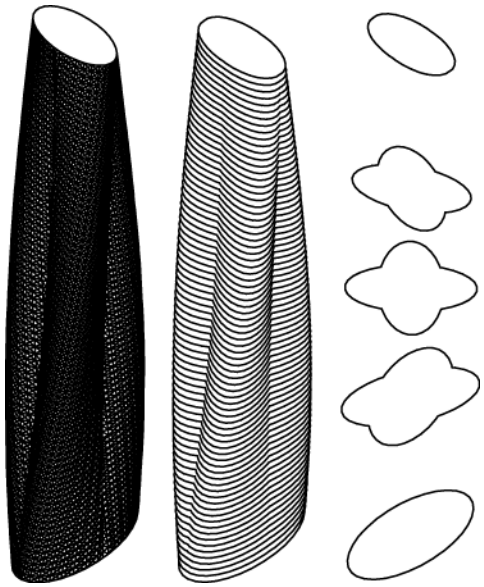

Figure 5.79d: PROG16a3, combined elliptical towers, with positive and negative rotation

Results for PROG16a3:

Area=903109 Perm=33341

Add function PROG16a3 using a copy of PROG16a1. Also add function PROG16e using a copy of PROG16b.

In this case each tower is rotated ninety degrees:

```
; first tower
(setq NumLevels 80)
(setq LevelHeight (* 12 10.0))
(setq NumSides 32)
(setq XRadStart (* 12 90.0))
(setq XRadEnd (* 12 60.0))
(setq YRadStart (* 12 45.0))
(setq YRadEnd (* 12 30.0))
(setq RotTotal 90.0)
(setq adist (* 12 0.0))
(prog16e)
; second tower
(setq RotTotal -90.0)
(prog16e)
```

Function PROG16e includes rotation of the floor plate as shown in PROG04.

Function PROG16e includes two executes of PROH16e, one for a positive rotation and the second for a negative rotation.

Completed function PROG16a3:

```
;-----------------------------------
(defun prog16a3 ()
 (graphscr)
 ; tower by polygon repeats
 ; elliptical form
```

```
 (setvar "CMDECHO" 0)
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get tower parameters
 (princ "\nProg16a3 - Tower")
 (setq TowerLayer "prog16a3")
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; turn off undo
 (command ".UNDO" "end")
 ; first tower
 (setq NumLevels 80)
 (setq LevelHeight (* 12 10.0))
 (setq NumSides 32)
 (setq XRadStart (* 12 90.0))
 (setq XRadEnd (* 12 60.0))
 (setq YRadStart (* 12 45.0))
 (setq YRadEnd (* 12 30.0))
 (setq RotTotal 90.0)
 (setq adist (* 12 0.0))
 (prog16e)
 ; second tower
 (setq RotTotal -90.0)
 (prog16e)
 ; combine towers
 (command ".UNION" "all" "")
 (setq obj1 (ssadd (entlast)))
 (command ".VIEW" "top")
 (command ".ZOOM" "e")
 ; do floor sections
 (setq tdata (prog16c))
 (setq tarea (nth 0 tdata))
 (setq tperm (nth 1 tdata))
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (princ "\n") (princ "Area=")
 (princ (rtos (/ tarea 144) 2 0))
 (princ "  ") (princ "Perm=")
 (princ (rtos (/ tperm 12) 2 0))
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 ; turn on history
 (setvar "SOLIDHIST" 0)
 ; turn on undo
 (command ".UNDO" "begin")
 (setvar "CMDECHO" 1)
 (princ)
)
;-----------------------------------
```

Completed function PROG16e:

```
;-----------------------------------
(defun prog16e ( / plist XRadInc XRad
 YRadInc YRad panginc pang xpt ypt
 npnt LevelElev RotInc Rotang)
 ; tower by polygon repeats
 ; elliptical form w/rotation
 ; set elev
 (setq LevelElev 0.0)
 ; start LOFT selection list
 (setq plist (ssadd))
 ; radius inc
 (setq XRadInc
   (/ (- XRadEnd XRadStart)
   NumLevels))
 (setq XRad XRadStart)
 ; radius inc
 (setq YRadInc
   (/ (- YRadEnd YRadStart)
   NumLevels))
 (setq YRad YRadStart)
 ; rotation inc
 (setq RotInc
   (/ RotTotal NumLevels))
 (setq Rotang 0.0)
 ; create each level
 (repeat (+ NumLevels 1)
  ; draw polygon
  (setq panginc (/ 360.0 NumSides))
  (setq pang (/ panginc 2.0))
  ; start polyline
  (command ".PLINE")
  (repeat (+ NumSides 1)
   ; compute point
   (setq xpt (+ (nth 0 cpnt)
     (* XRad (sin (dtr pang)))))
   (setq ypt (+ (nth 1 cpnt)
     (* YRad (cos (dtr pang)))))
   ; rotate point
   (setq rxpt
     (- (* xpt (cos (dtr Rotang)))
     (* ypt (sin (dtr Rotang)))))
   (setq rypt
     (+ (* xpt (sin (dtr Rotang)))
     (* ypt (cos (dtr Rotang)))))
   (setq npnt
     (list rxpt rypt LevelElev))
   ; add to polyline
   (command npnt)
   ; inc ang
   (setq pang (+ pang panginc))
  )
  ; close polyline
  (command "c")
  (command ".ZOOM" "e")
  ; add to LOFT list
  (setq plist
   (ssadd (entlast) plist))
  ; inc radius
  (setq XRad (+ XRad XRadInc))
  (setq YRad (+ YRad YRadInc))
  ; inc elev
  (setq LevelElev
    (+ LevelElev LevelHeight))
  ; inc rotation
  (setq Rotang (+ Rotang Rotinc))
 )
 (command ".ZOOM" "e")
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set LOFTNORMALS =1 for smooth
 ; =0 for ruled
 (setvar "LOFTNORMALS" 0)
 ; set LOFTPARAM to default
 (setvar "LOFTPARAM" 7)
 ; loft sections
 (command ".LOFT" plist "" "")
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 (princ)
)
;-----------------------------------
```

Develop additional concepts to combine individual towers of the same configuration and of differing configuration.

Tower model floor plans generated from combination of fixed shapes

Many of the examples demonstrated a single shape that could be computed by a simple series of points. Many more complex shapes could be considered if they were predefined in some fashion.

For function PROG17a we consider the case where we develop a typical floor plan as a BLOCK. In this case the BLOCK consists of unioned regions of simple geometric shapes; circles, rectangles, and ellipses. Each BLOCK is predefined at actual size, named, and an

insert point defined. The insert point will be used to location, scales, and rotate the floor plan.

The three examples demonstrate a variety of basic floor plates.

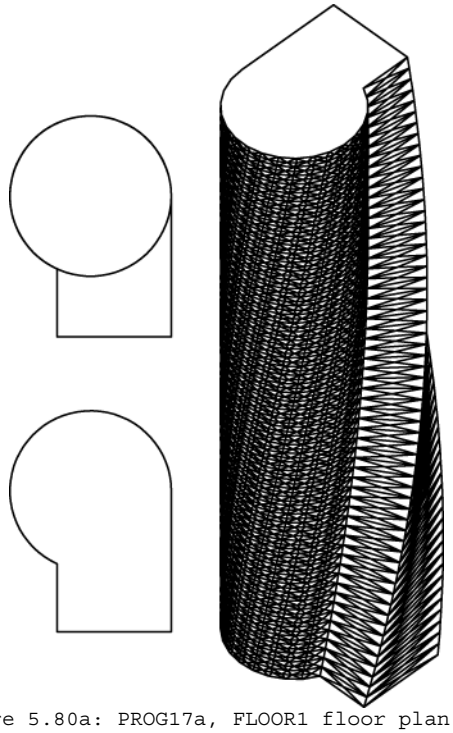

Figure 5.80a: PROG17a, FLOOR1 floor plan

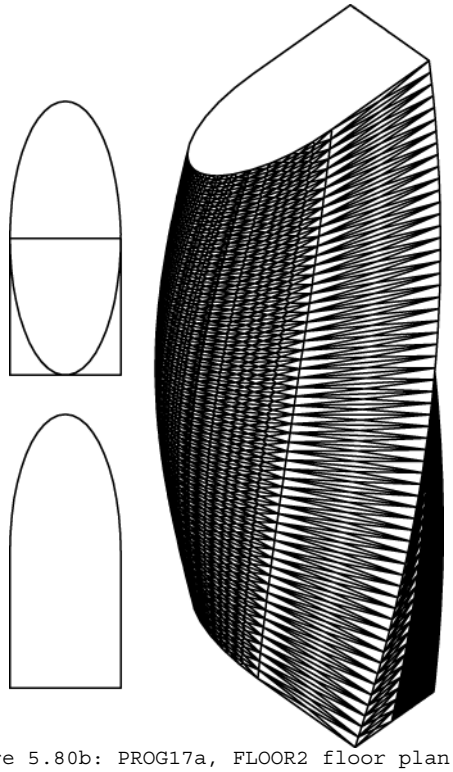

Figure 5.80b: PROG17a, FLOOR2 floor plan

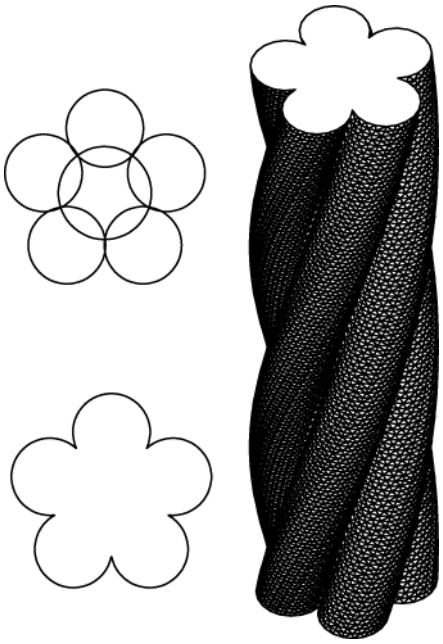

Figure 5.80c: PROG17a, FLOOR3 floor plan

Add function PROG17a using a copy of PROG04a. Replace the section creating a polygon with the insertion of a BLOCK.

Add input for the floor plan BLOCK name:

```
(setq FloorBlock
   (getstring
   "\nEnter floor block name: "))
```

Input for the polygon and its radius is removed. Computations for the radius increment are also removed.

The section that computes the polygon points and rotated them is replaced with the insertion and rotation of the BLOCK:

```
; get block
(setq ipnt (list
 (nth 0 cpnt) (nth 1 cpnt)
 LevelElev))
(command ".INSERT" FloorBlock
 ipnt "1" "1" Rotang)
(command ".ZOOM" "e")
; explode and set layer
(command ".EXPLODE" "l")
(command ".CHPROP" "p" ""
 "layer" TowerLayer "")
; area
(if (<= nlevel NumLevels) (progn
 (command ".AREA" "o" "last")
 (setq tarea
  (+ tarea (getvar "AREA")))
 (setq tperm
  (+ tperm (getvar "PERIMETER")))
))
; add to LOFT list
(setq plist
 (ssadd (entlast) plist))
```

Note the command to INSERT the BLOCK, how the BLOCK layer is changed to the current one, and the computation of the area. A level counter is included so the area of the top floor, the roof is not counted.

In this function the INSERT command:

```
(command ".INSERT" FloorBlock
 ipnt "1" "1" Rotang)
```

sets the scale of the BLOCK to one. This could easily be modified to include a scale factor, inplemneted similar to the way the rotation is.

Sample script file for PROG17a:

```
;-----------------------------------
(prog17a)
; Layer name:
LAYER17a_1
; Floor block:
FLOOR01
; number of levels:
60
; level height:
10'
; rotation:
90
;-----------------------------------
```

Results for FLOOR01 example:

```
Levels: 60
Area: 957115 sqft   Perm: 29845 ft
```

The other two examples use blocks FLOOR02 and FLOOR03.

Results for FLOOR02 and FLOOR03 examples:

```
Levels: 60
Area: 956202 sqft   Perm: 32236 ft

Levels: 60
Area: 948933 sqft   Perm: 35946 ft
```

Completed function PROG17a:

```
;-----------------------------------
(defun prog17a ()
 (graphscr)
 ; tower by block repeats
 ; twist
 (setvar "CMDECHO" 0)
 ; turn off undo
 (command ".UNDO" "c" "n")
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get curve parameters
 (princ "\nProg17a - Tower")
 (setq TowerLayer
   (getstring
   "\nEnter layer name: "))
 (setq FloorBlock
   (getstring
   "\nEnter floor block name: "))
 (setq NumLevels
   (getint
   "\nEnter number of levels: "))
 (setq LevelHeight
   (getdist
   "\nEnter level height: "))
 (setq RotTotal
   (getreal
   "\nEnter total rotation
   angle: "))
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; set elev
 (setq LevelElev 0.0)
 ; area and perm
 (setq tarea 0.0)
 (setq tperm 0.0)
 ; start LOFT selection list
 (setq plist (ssadd))
 ; rotation inc
 (setq RotInc
   (/ RotTotal NumLevels))
 (setq Rotang 0.0)
 ; create each level
 (setq nlevel 1)
 (repeat (+ NumLevels 1)
  ; get block
  (setq ipnt (list
   (nth 0 cpnt) (nth 1 cpnt)
   LevelElev))
  (command ".INSERT" FloorBlock
   ipnt "1" "1" Rotang)
  (command ".ZOOM" "e")
  ; explode and set layer
  (command ".EXPLODE" "l")
  (command ".CHPROP" "p" ""
   "layer" TowerLayer "")
  ; area
  (if (<= nlevel NumLevels) (progn
   (command ".AREA" "o" "last")
   (setq tarea
    (+ tarea (getvar "AREA")))
   (setq tperm
    (+ tperm (getvar "PERIMETER")))
  ))
  ; add to LOFT list
  (setq plist
   (ssadd (entlast) plist))
  ; inc elev
  (setq LevelElev
    (+ LevelElev LevelHeight))
  ; inc rotation
  (setq Rotang (+ Rotang Rotinc))
  ; inc level
  (setq nlevel (+ nlevel 1))
 )
 (command ".ZOOM" "e")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set LOFTNORMALS =1 for smooth
 ; =0 for ruled
 (setvar "LOFTNORMALS" 0)
 ; set LOFTPARAM to default
 (setvar "LOFTPARAM" 7)
 ; loft sections
 (command ".LOFT" plist "" "")
 ; turn on history
 (setvar "SOLIDHIST" 1)
 (setvar "CMDECHO" 1)
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 ; turn on undo
 (command ".UNDO" "a")
 ; area and perm
 (princ "\nLevels: ")
  (princ NumLevels)
  (princ "   Area: ")
  (princ (rtos (/ tarea 144) 2 0))
  (princ " sqft   Perm: ")
  (princ (rtos (/ tperm 12) 2 0))
  (princ " ft")
 (princ)
)
;-----------------------------------
```

The floor plans as a fixed predefined shapes can also be used as individual sections within the tower. One floor plan can transition to another. The transition can also included rotation and scaling.

In the first example FLOOR4 is a circle floor plan.

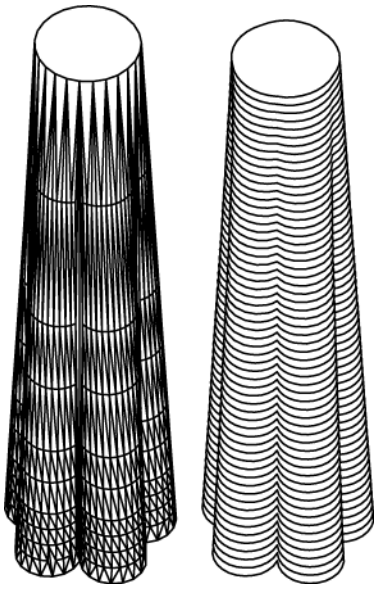

Figure 5.80d: PROG17b, FLOOR3 and FLOOR4 scaled

Sample script file for PROG17b:

```
;-----------------------------------
(prog17b)
; Layer name:
LAYER17b_1
; Number of blocks:
2
; Block specs:
( "FLOOR03" 0  1.0 0.0 )
( "FLOOR04" 60 0.6 0.0 )
; level height
10'
;-----------------------------------
```

Results for FLOOR3 and FLOOR4 example:

Levels: 60
Area: 674969 sqft   Perm: 26066 ft

In the second example, FLOOR3 is rotated and scaled.

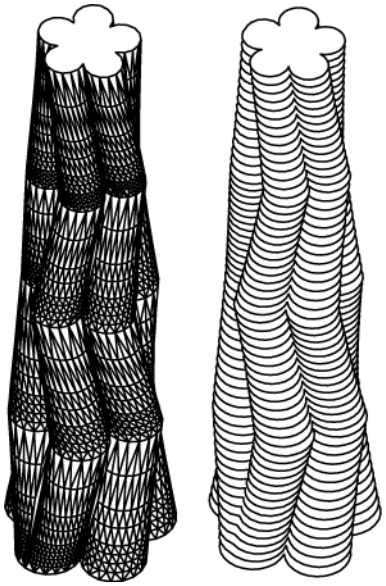

Figure 5.80e: PROG17b, FLOOR3 floor plan scaled and rotated

Sample script file for PROG17b:

```
;-----------------------------------
(prog17b)
; Layer name:
LAYER17b_2
; Number of blocks:
5
; Block specs:
( "FLOOR03" 0  1.0 0.0  )
( "FLOOR03" 15 0.9 36.0 )
( "FLOOR03" 30 0.8 0.0 )
( "FLOOR03" 45 0.7 36.0 )
( "FLOOR03" 60 0.6 0.0 )
; level height
10'
;-----------------------------------
```

Results for FLOOR3 example:

Levels: 60
Area: 603554 sqft   Perm: 28159 ft

Add function PROG17b using a copy of PROG17a. The input for the floor plan changes to a list:

```
( "FLOOR03" 0  1.0 0.0 )
```

Included is the block name, floor level, scale and rotation angle.

Note the use of the read function to convert the string into a list:

```
(setq nBlocks
  (getint
  "\nEnter number of blocks: "))
(setq lblocks (list ))
(repeat nBlocks
 (setq spec (read
  (getstring
  "\nEnter block specs:")))
 (setq lBlocks
  (append lBlocks (list spec)))
)
```

The function changes from generating individual floor plates to including only floor plates hewn the sections change. The insertion of the BLOCK is modified to include the input list and a series of sections.

Review the placing of the floor plates:

```
; start LOFT selection list
 (setq plist (ssadd))
 ; get blocks
 (setq nspec 0)
 (repeat nBlocks
  (setq spec (nth nspec lBlocks))
  ; get specs
  (setq FloorBlock (nth 0 spec))
  (setq NumLevels (nth 1 spec))
  (setq LevelElev
   (* NumLevels LevelHeight))
  ; add top elev
  (if (= nspec (- nBlocks 1))
   (setq LevelElev
    (+ LevelElev LevelHeight)))
  (setq XYScale (nth 2 spec))
  (setq Rotang (nth 3 spec))
  ; place block
  (setq ipnt (list
   (nth 0 cpnt) (nth 1 cpnt)
   LevelElev))
  (command ".INSERT" FloorBlock
   ipnt XYScale XYScale Rotang)
  (command ".ZOOM" "e")
  ; explode and set layer
  (command ".EXPLODE" "l")
  (command ".CHPROP" "p" ""
   "layer" TowerLayer "")
```

```
  ; add to LOFT list
  (setq plist
   (ssadd (entlast) plist))
  ; next block
  (setq nspec (+ nspec 1))
 )
```

Completed function PROG17b:

```
;-----------------------------------
(defun prog17b ()
 (graphscr)
 ; tower by block repeats
 (setvar "CMDECHO" 0)
 ; turn off undos
 (command ".UNDO" "c" "n")
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get curve parameters
 (princ "\nProg17b - Tower")
 (setq TowerLayer
   (getstring
   "\nEnter layer name: "))
 (setq nBlocks
   (getint
   "\nEnter number of blocks: "))
 (setq lblocks (list ))
 (repeat nBlocks
  (setq spec (read
   (getstring
   "\nEnter block specs:")))
  (setq lBlocks
   (append lBlocks (list spec)))
 )
 (setq LevelHeight
   (getdist
   "\nEnter level height: "))
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; start LOFT selection list
 (setq plist (ssadd))
 ; get blocks
 (setq nspec 0)
 (repeat nBlocks
  (setq spec (nth nspec lBlocks))
  ; get specs
  (setq FloorBlock (nth 0 spec))
  (setq NumLevels (nth 1 spec))
  (setq LevelElev
   (* NumLevels LevelHeight))
  ; add top elev
  (if (= nspec (- nBlocks 1))
   (setq LevelElev
    (+ LevelElev LevelHeight)))
  (setq XYScale (nth 2 spec))
  (setq Rotang (nth 3 spec))
  ; place block
  (setq ipnt (list
   (nth 0 cpnt) (nth 1 cpnt)
   LevelElev))
  (command ".INSERT" FloorBlock
   ipnt XYScale XYScale Rotang)
  (command ".ZOOM" "e")
  ; explode and set layer
  (command ".EXPLODE" "l")
  (command ".CHPROP" "p" ""
   "layer" TowerLayer "")
  ; add to LOFT list
  (setq plist
   (ssadd (entlast) plist))
  ; next block
  (setq nspec (+ nspec 1))
 )
 (command ".ZOOM" "e")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set LOFTNORMALS =1 for smooth
 ; =0 for ruled
 (setvar "LOFTNORMALS" 0)
 ; set LOFTPARAM to default
 (setvar "LOFTPARAM" 7)
 ; loft sections
 (command ".LOFT" plist "" "")
 (setq obj1 (ssadd (entlast)))
 ; do floor sections
 (setq tdata (prog16c))
 (setq tarea (nth 0 tdata))
 (setq tperm (nth 1 tdata))
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 ; area and perm
 (princ "\nLevels: ")
  (princ NumLevels)
  (princ "   Area: ")
  (princ (rtos (/ tarea 144) 2 0))
  (princ " sqft   Perm: ")
  (princ (rtos (/ tperm 12) 2 0))
  (princ " ft")
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 ; turn on history
 (setvar "SOLIDHIST" 0)
 ; turn on undo
 (command ".UNDO" "a")
 (setvar "CMDECHO" 1)
 (princ)
)
;-----------------------------------
```

Tower model floor plans generated from combination of individual shapes

The previous example included fixed floor plates that could be placed, rotated, and scaled. The entire floor plate was a single shape.

This example demonstrates how a set of basic shapes could be defined as a single floor plate but still retain the ability to vary each individual shape.

The first example includes a combination of a static rectangle and a contracting circle.

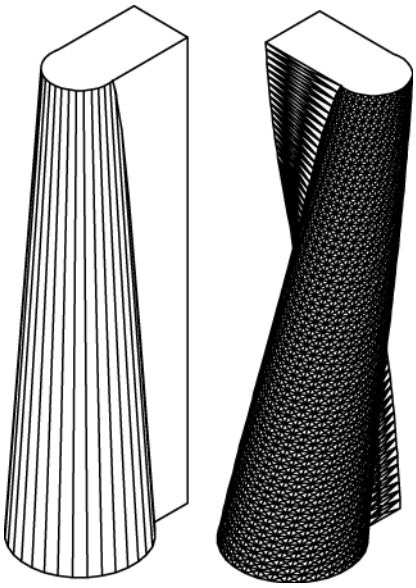

Figure 5.81a: PROG18a, rectangle and circle, without/with rotation

Results:

```
Levels: 60
Area: 1019317 sqft   Perm: 31348 ft
```

Sample script file for PROG18a:

```
;-----------------------------------
(prog18a)
; Layer name:
LAYER18a_1
; levels
60
; level height
10'
; shapes
2
( (-60 0) 32 80 80 41 41 0 )
( (0 0) 0  120 80 120 80 0 )
; rotation
0
;-----------------------------------
```

The second example is a combination of a static ellipse and a contracting circle.

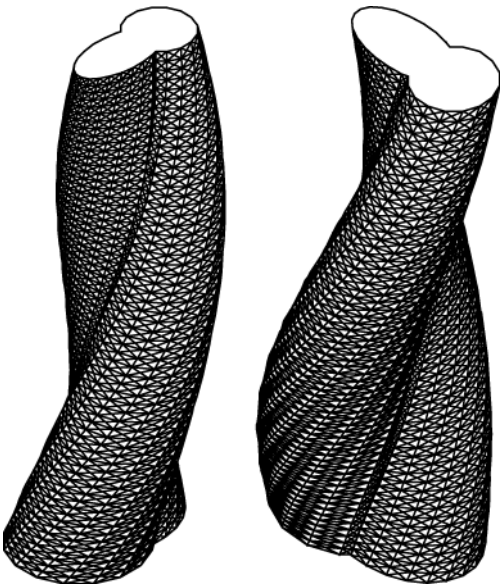

Figure 5.81b: PROG18a, ellipse and circle, isometric view

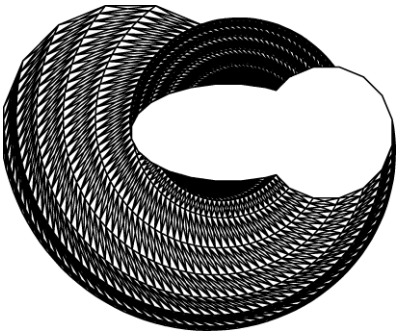

Figure 5.81c: PROG18a, ellipse and circle, plan view

Sample script file for PROG18a:

```
;-----------------------------------
(prog18a)
; Layer name:
LAYER18a_2
; levels
60
; level height
10'
; shapes
2
( (-60 0) 16 80 80 41 41 )
( (0 0) 24 80 60 60 30 )
; rotation
180
;-----------------------------------
```

Results:

```
Levels: 60
Area: 1016034 sqft   Perm: 30074 ft
```

The third example is a combination of a static circle and two contracting ellipses.

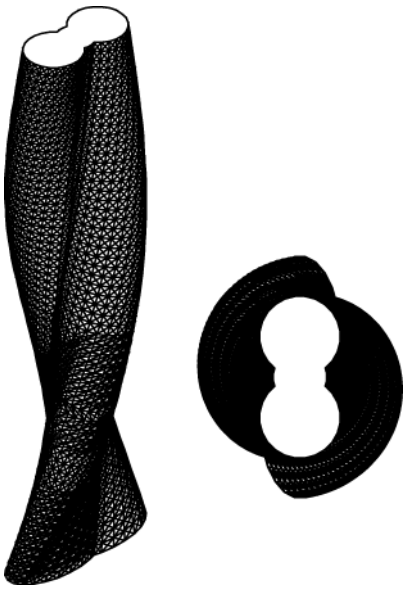

Figure 5.81d: PROG18a, circle and two ellipses

Sample script file for PROG18a:

```
;-----------------------------------
(prog18a)
; Layer name:
LAYER18a_3
; levels
60
; level height
10'
; shapes
3
( (-30 0) 24 60 30 30 30 )
( ( 30 0) 24 60 30 30 30 )
( (  0 0) 24 20 20 20 20 )
; rotation
180
;-----------------------------------
```

Results:

```
Levels: 60
Area: 447028 sqft   Perm: 21859 ft
```

Add function PROG18a using a copy of PROG17a.

The input is changed from a BLOCK name to a list of shapes:

```
(setq nShapes
  (getint
  "\nEnter number of shape specs:"))
```

```
(setq lshapes (list ))
(repeat nShapes
 (setq spec (read
  (getstring
  "\nEnter shape specs:")))
 (setq lshapes
  (append (list spec) lshapes))
)
```

Each shape is defined by the list, for example:

```
( (-60 0) 32 80 80 41 41 )
( (0 0) 0  120 80 120 80 )
```

The list defines a polygon including its center location, XY relative to global (0,0), number of sides, starting X and Y radius, and ending X and Y radius. This covers circles, polygons and ellipses. If the number of sides is zero, then a rectangle is assumed and the next parameters are start length and width, and end length and width.

These set of shapes are combined into a single floor plate, located at the appropriate height and rotated, if specified.

Each floor plate is collected into a selection list and then LOFTed as before. Since each floor is individual computed, area and perimeters can be computed as they are generated.

Review function PROG18a, how the shape list parameters are extracted, how the scale of each shape is computed based on floor height, the union of the individual shapes, and its rotation.

Completed function PROG18a:

```
;-----------------------------------
(defun prog18a ()
 (graphscr)
 ; tower by unioned shapes repeats
 ; twist
 (setvar "CMDECHO" 0)
 ; turn off undo
 (command ".UNDO" "c" "n")
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get curve parameters
 (princ "\nProg18a - Tower")
 (setq TowerLayer
   (getstring
   "\nEnter layer name: "))
 (setq NumLevels
   (getint
   "\nEnter number of levels: "))
 (setq LevelHeight
   (getdist
   "\nEnter level height: "))
 (setq nShapes
   (getint
   "\nEnter number of shape specs:"))
 (setq lshapes (list ))
 (repeat nShapes
  (setq spec (read
   (getstring
   "\nEnter shape specs:")))
  (setq lshapes
   (append (list spec) lshapes))
 )
 (setq RotTotal
   (getreal
   "\nEnter total rotation
   angle: "))
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; set elev
 (setq LevelElev 0.0)
 ; area and perm
 (setq tarea 0.0)
 (setq tperm 0.0)
 ; start LOFT selection list
 (setq plist (ssadd))
 ; rotation inc
 (setq RotInc
   (/ RotTotal NumLevels))
 (setq Rotang 0.0)
 ; create each level
 (setq nlevel 1)
 (repeat (+ NumLevels 1)
  ; union list
  (setq ulist (ssadd))
  ; get shape
  (setq ns 0)
  (repeat nShapes
   (setq spec (nth ns lshapes))
   (setq cpt (nth 0 spec))
   (setq cpt (list
    (* (nth 0 cpt) 12.0)
    (* (nth 1 cpt) 12.0)))
   (setq NumSides (nth 1 spec))
   ; rectangle
   (if (= NumSides 0) (progn
    (setq rslen (* (nth 2 spec) 12.0))
    (setq rswid (* (nth 3 spec) 12.0))
    (setq relen (* (nth 4 spec) 12.0))
    (setq rewid (* (nth 5 spec) 12.0))
    (setq rlen
     (+ rslen
     (* (/ (- relen rslen)
     (+ NumLevels 1)) nlevel)))
    (setq rwid
     (+ rswid
     (* (/ (- rewid rswid)
     (+ NumLevels 1)) nlevel)))
    (setq pnt1 (list
     (- (nth 0 cpt) (/ rlen 2))
     (- (nth 1 cpt) (/ rwid 2))
     LevelElev))
    (setq pnt2 (list
     (+ (nth 0 cpt) (/ rlen 2))
     (+ (nth 1 cpt) (/ rwid 2))
     LevelElev))
    (command ".RECTANGLE" pnt1 pnt2)
   ))
   ; polygon
   (if (> NumSides 0) (progn
    (setq psxrad (* (nth 2 spec) 12.0))
    (setq psyrad (* (nth 3 spec) 12.0))
    (setq pexrad (* (nth 4 spec) 12.0))
    (setq peyrad (* (nth 5 spec) 12.0))
    (setq xRad
     (+ psxrad
     (* (/ (- pexrad psxrad)
     (+ NumLevels 1)) nlevel)))
    (setq yRad
     (+ psyrad
     (* (/ (- peyrad psyrad)
     (+ NumLevels 1)) nlevel)))
    ; start polyline
    (setq panginc (/ 360.0 NumSides))
    (setq pang (/ panginc 2.0))
    (command ".PLINE")
    (repeat (+ NumSides 1)
     ; compute point
     (setq xpt (+ (nth 0 cpt)
      (* xRad (sin (dtr pang)))))
     (setq ypt (+ (nth 1 cpt)
      (* yRad (cos (dtr pang)))))
     (setq npnt
       (list xpt ypt LevelElev))
     ; add to polyline
     (command npnt)
     ; inc ang
```

```
     (setq pang (+ pang panginc))
    )
    ; close polyline
    (command "c")
   ))
   (command ".ZOOM" "e")
   ; region
   (command ".REGION" "last" "")
   ; add to union list
   (setq ulist
    (ssadd (entlast) ulist))
   ; inc shapes
   (setq ns (+ ns 1))
  )
  ; union shapes
  (if (> nShapes 1)
   (command ".UNION" ulist ""))
  ; rotate
  (command ".ROTATE" "last" ""
   cpnt Rotang)
  ; area
  (if (<= nlevel NumLevels) (progn
   (command ".AREA" "o" "last")
   (setq tarea
    (+ tarea (getvar "AREA")))
   (setq tperm
    (+ tperm (getvar "PERIMETER")))
  ))
  ; add to LOFT list
  (setq plist
   (ssadd (entlast) plist))
  ; inc elev
  (setq LevelElev
    (+ LevelElev LevelHeight))
  ; inc rotation
  (setq Rotang (+ Rotang Rotinc))
  ; inc levels
  (setq nlevel (+ nlevel 1))
 )
 (command ".ZOOM" "e")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set LOFTNORMALS =1 for smooth
 ; =0 for ruled
 (setvar "LOFTNORMALS" 0)
 ; set LOFTPARAM to default
 (setvar "LOFTPARAM" 7)
 ; loft sections
 (command ".LOFT" plist "" "")
 ; turn on history
 (setvar "SOLIDHIST" 1)
 (setvar "CMDECHO" 1)
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 ; turn on undo
 (command ".UNDO" "a")
 ; area and perm
 (princ "\nLevels: ")
  (princ NumLevels)
  (princ "   Area: ")
  (princ (rtos (/ tarea 144) 2 0))
  (princ " sqft   Perm: ")
  (princ (rtos (/ tperm 12) 2 0))
  (princ " ft")
 (princ)
)
;-----------------------------------
```

Tower model with incremental rotation of floors

Floor plate rotation in the previous examples was smoothly applied at each level. The floors can also be rotated independently of each other, in the a fan manner, individually or in blocks.

The first example using FLOOR1, groups floors into blocks of five levels.

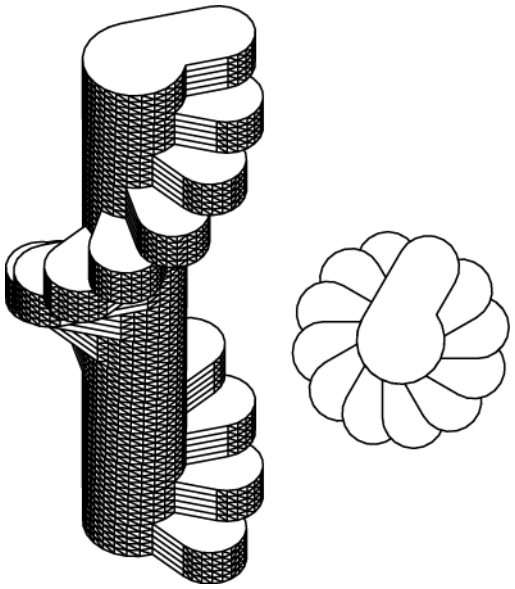

Figure 5.82a: PROG19a, five levels per floor segment

Results:

Levels: 75
Area: 1409187 sqft   Perm: 40945 ft

Sample script files for PROG19a:

```
;-----------------------------------
(prog19a)
; Layer name:
LAYER19a_1
; block name:
FLOOR01
; number of blocks:
15
; levels per block:
5
; level height:
10'
; block rotation
30.0
;-----------------------------------
```

The second example using FLOOR1, groups floors into blocks of single levels.

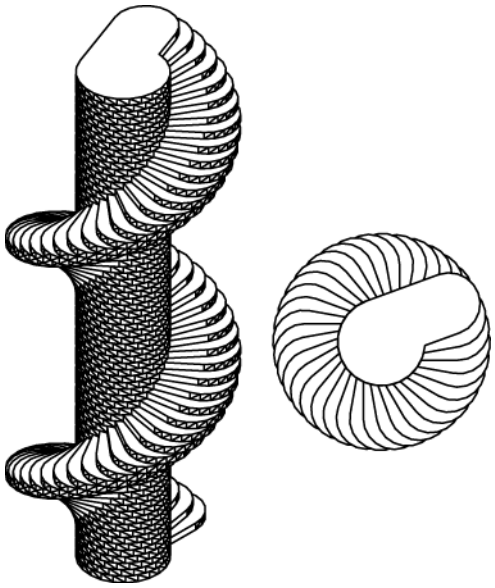

Figure 5.82b: PROG19a, individual floor segments

Sample script files for PROG19a:

```
;-----------------------------------
(prog10a)
; Layer name:
LAYER19a_1a
; block name:
FLOOR01
; number of blocks:
75
; levels per block:
1
; level height:
10'
; block rotation
10.0
;-----------------------------------
```

The third example using FLOOR2, groups the floors inot blocks ofthree levels.

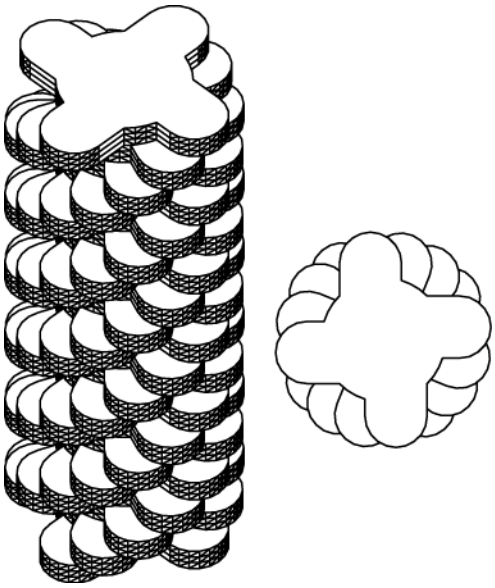

Figure 5.82c: PROG19a,

Results:

Levels: 75
Area: 3211668 sqft Perm: 73080 ft

Sample script file for PROG19a:

```
;-----------------------------------
(prog19a)
; Layer name:
LAYER19a_2
; block name:
FLOOR02
; number of blocks:
25
; levels per block:
3
; level height:
10'
; block rotation
22.5
;-----------------------------------
```

Add function PROG19a using PROG17a, where the floor plate is a fixed shape predefined as a BLOCK.

Review function PROG19a. The generation of the floors is separated into extruding the floor plate by floor with a rotation between groups of floors. Each floor section is repeated within a group. Since the floor plate is extruded individually, the lofting is not needed.

Completed function PROG19a:

```
;-----------------------------------
(defun prog19a ()
 (graphscr)
 ; tower by block repeats
 ; twist with blocks of floors
 (setvar "CMDECHO" 0)
 ; turn off undos
 (command ".UNDO" "c" "n")
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get curve parameters
 (princ "\nProg19a - Tower")
 (setq TowerLayer
   (getstring
   "\nEnter layer name: "))
 (setq FloorBlock
   (getstring
   "\nEnter floor block name: "))
 (setq NumBlocks
   (getint
   "\nEnter number of blocks: "))
 (setq NumLevels
   (getint
   "\nEnter levels per block: "))
 (setq LevelHeight
   (getdist
   "\nEnter level height: "))
 (setq RotInc
   (getreal
   "\nEnter rotation angle inc: "))
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; turn off history
 (setvar "SOLIDHIST" 0)
 ; set DELOBJ to delete the sections
 (setvar "DELOBJ" 1)
 ; set elev
 (setq LevelElev 0.0)
 ; area and perm
 (setq tarea 0.0)
 (setq tperm 0.0)
 ; rotation
 (setq Rotang 0.0)
 ; create each level
 (repeat NumBlocks
  (repeat NumLevels
   ; get block
   (setq ipnt (list
    (nth 0 cpnt) (nth 1 cpnt)
    LevelElev))
   (command ".INSERT" FloorBlock
    ipnt "1" "1" Rotang)
   (command ".ZOOM" "e")
   ; explode and set layer
   (command ".EXPLODE" "l")
   (command ".CHPROP" "p" ""
    "layer" TowerLayer "")
   ; area
   (command ".AREA" "o" "last")
   (setq tarea
    (+ tarea (getvar "AREA")))
   (setq tperm
    (+ tperm (getvar "PERIMETER")))
   ; extrude
```

```
   (command ".EXTRUDE" "last" ""
    LevelHeight )
   ; inc elev
   (setq LevelElev
    (+ LevelElev LevelHeight))
  )
  ; inc rotation
  (setq Rotang (+ Rotang RotInc))
 )
 (command ".ZOOM" "e")
 ; turn on history
 (setvar "SOLIDHIST" 1)
 (setvar "CMDECHO" 1)
 (command ".VIEW" "swiso")
 (command ".ZOOM" "e")
 (command ".HIDE")
 ; turn on undo
 (command ".UNDO" "a")
 ; area and perm
 (princ "\nLevels: ")
  (princ (* NumBlocks NumLevels))
  (princ "   Area: ")
  (princ (rtos (/ tarea 144) 2 0))
  (princ " sqft   Perm: ")
  (princ (rtos (/ tperm 12) 2 0))
  (princ " ft")
 (princ)
)
;-----------------------------------
```

The grouping of floor can also be developed using the floor plate created by individually changing shapes.

The first example groups floor consisting of a combination of a static circle and one that is contracting rotated.

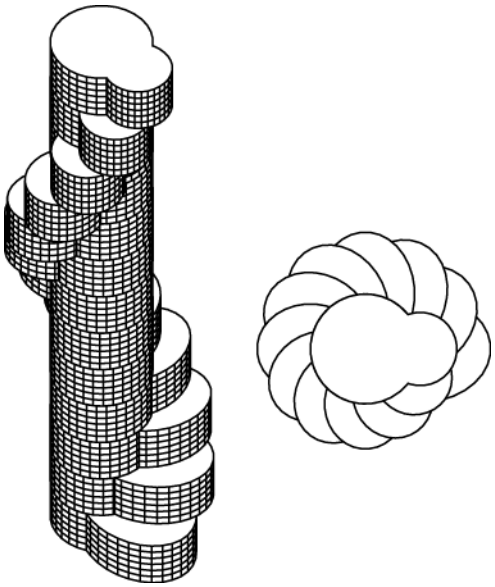

Figure 5.83a: PROG20a, floor groups combining two circles with rotation

Results:

```
Levels: 78
Area: 1292773 sqft   Perm: 37883 ft
```

Sample script files for PROG20a:

```
;-----------------------------------
(prog20a)
; Layer name:
LAYER20a_1
; number of blocks:
13
; levels per block:
6
; level height:
10'
; shapes
2
( (0 0) 32 60 60 60 60 0 )
( (60 0) 32 80 60 40 40 0 )
; block rotation:
30.0
;-----------------------------------
```

The second example groups floor consisting of a combination of a static circle and one that is contracting not rotated.

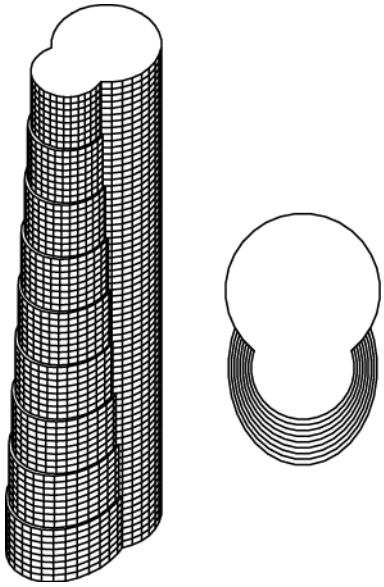

Figure 5.83b: PROG20a, floor groups combining two circles without rotation

Results:

```
Levels: 72
Area: 1186135 sqft   Perm: 34866 ft
```

Sample script files for PROG20a:

```
;-----------------------------------
(prog20a)
; Layer name:
LAYER20a_1a
; number of blocks:
9
; levels per block:
8
; level height:
10'
; shapes
2
( (0 0) 32 60 60 60 60 0 )
( (60 0) 32 80 60 40 40 0 )
; block rotation:
0.0
;-----------------------------------
```

Function PROG20a is based on PROG18a and PROG19a.

Completed function PROG20a:

```
;-----------------------------------
(defun prog20a ()
 (graphscr)
 ; tower by unioned shapes repeats
 ; twist
 (setvar "CMDECHO" 0)
 ; turn off undo
 (command ".UNDO" "c" "n")
 ; center point
```

```
(setq cpnt (list 0.0 0.0 0.0))
; get curve parameters
(princ "\nProg20a - Tower")
(setq TowerLayer
  (getstring
  "\nEnter layer name: "))
(setq NumBlocks
  (getint
  "\nEnter levels per blocks: "))
(setq NumLevels
  (getint
  "\nEnter levels per blocks: "))
(setq LevelHeight
  (getdist
  "\nEnter level height: "))
(setq nShapes
  (getint
  "\nEnter number of shape specs:"))
(setq lshapes (list ))
(repeat nShapes
 (setq spec (read
  (getstring
  "\nEnter shape specs:")))
 (setq lshapes
  (append (list spec) lshapes))
)
(setq RotInc
  (getreal
  "\nEnter rotation angle inc: "))
; clear layer
(command ".LAYER" "THAW" "*"
  "ON" "*" "")
(command ".LAYER"
  "MAKE" TowerLayer "")
(command ".LAYER" "SET" 0 "")
(command ".LAYER" "OFF" "@*"
  "FREEZE" "@*" "")
(command ".LAYER" "THAW" TowerLayer
  "ON" TowerLayer
  "SET" TowerLayer "")
(command ".ERASE" "ALL" "")
; turn off history
(setvar "SOLIDHIST" 0)
; set DELOBJ to delete the sections
(setvar "DELOBJ" 1)
; set elev
(setq LevelElev 0.0)
; area and perm
(setq tarea 0.0)
(setq tperm 0.0)
; rotation
(setq Rotang 0.0)
; create each level
(setq nlevel 1)
(repeat NumBlocks
 (repeat NumLevels
  ; union list
  (setq ulist (ssadd))
  ; get shape
  (setq ns 0)
  (repeat nShapes
   (setq spec (nth ns lshapes))
   (setq cpt (nth 0 spec))
   (setq cpt (list
    (* (nth 0 cpt) 12.0)
    (* (nth 1 cpt) 12.0)))
   (setq NumSides (nth 1 spec))
   ; rectangle
   (if (= NumSides 0) (progn
    (setq rslen (* (nth 2 spec) 12.0))
    (setq rswid (* (nth 3 spec) 12.0))
    (setq relen (* (nth 4 spec) 12.0))
    (setq rewid (* (nth 5 spec) 12.0))
    (setq rlen
     (+ rslen
     (* (/ (- relen rslen)
     (+ NumLevels 1)) nlevel)))
    (setq rwid
     (+ rswid
     (* (/ (- rewid rswid)
     (+ NumLevels 1)) nlevel)))
    (setq pnt1 (list
     (- (nth 0 cpt) (/ rlen 2))
     (- (nth 1 cpt) (/ rwid 2))
     LevelElev))
    (setq pnt2 (list
     (+ (nth 0 cpt) (/ rlen 2))
     (+ (nth 1 cpt) (/ rwid 2))
     LevelElev))
    (command ".RECTANGLE" pnt1 pnt2)
   ))
   ; polygon
   (if (> NumSides 0) (progn
    (setq psxrad (* (nth 2 spec) 12.0))
    (setq psyrad (* (nth 3 spec) 12.0))
    (setq pexrad (* (nth 4 spec) 12.0))
    (setq peyrad (* (nth 5 spec) 12.0))
    (setq xRad
     (+ psxrad
     (* (/ (- pexrad psxrad)
     NumBlocks) nlevel)))
    (setq yRad
     (+ psyrad
     (* (/ (- peyrad psyrad)
     NumBlocks) nlevel)))
    ; start polyline
    (setq panginc (/ 360.0 NumSides))
    (setq pang (/ panginc 2.0))
    (command ".PLINE")
    (repeat (+ NumSides 1)
     ; compute point
     (setq xpt (+ (nth 0 cpt)
      (* xRad (sin (dtr pang)))))
     (setq ypt (+ (nth 1 cpt)
      (* yRad (cos (dtr pang)))))
     (setq npnt
      (list xpt ypt LevelElev))
     ; add to polyline
     (command npnt)
     ; inc ang
     (setq pang (+ pang panginc))
    )
    ; close polyline
    (command "c")
   ))
   (command ".ZOOM" "e")
   ; region
   (command ".REGION" "last" "")
   ; add to union list
   (setq ulist
    (ssadd (entlast) ulist))
   ; inc shapes
   (setq ns (+ ns 1))
  )
  ; union shapes
  (if (> nShapes 1)
   (command ".UNION" ulist ""))
  ; rotate
  (command ".ROTATE" "last" ""
   cpnt Rotang)
  ; area
  (command ".AREA" "o" "last")
  (setq tarea
   (+ tarea (getvar "AREA")))
  (setq tperm
   (+ tperm (getvar "PERIMETER")))
  ; extrude
  (command ".EXTRUDE" "last" ""
   LevelHeight )
  ; inc elev
  (setq LevelElev
   (+ LevelElev LevelHeight))
 )
 ; inc level
 (setq nlevel (+ nlevel 1))
 ; inc rotation
 (setq Rotang (+ Rotang RotInc))
)
(command ".ZOOM" "e")
; turn on history
(setvar "SOLIDHIST" 1)
(setvar "CMDECHO" 1)
(command ".VIEW" "swiso")
(command ".ZOOM" "e")
(command ".HIDE")
; turn on undo
(command ".UNDO" "a")
; area and perm
```

```
 (princ "\nLevels: ")
  (princ (* NumBlocks NumLevels))
  (princ "   Area: ")
  (princ (rtos (/ tarea 144) 2 0))
  (princ " sqft   Perm: ")
  (princ (rtos (/ tperm 12) 2 0))
  (princ " ft")
 (princ)
)
;-----------------------------------
```

Tower models with floor plan coverted to curves

The ability to make a solid model from individual floor sections from basic shapes allows us to not have to create every point. In the cases where we do compute a section point-by-point, these sections can be translated into a series of arcs, curve fit, or a spline.

To demonstrate this variation use a copy of function PROG13 from this CH05C.LSP, morphing sections based on a top and bottom series of points.

The first example demonstrates the morphing between a top and bottom set of points, no rotation. In this example the top set has an extra midpoint so the bottom set of points can be morphed to it.

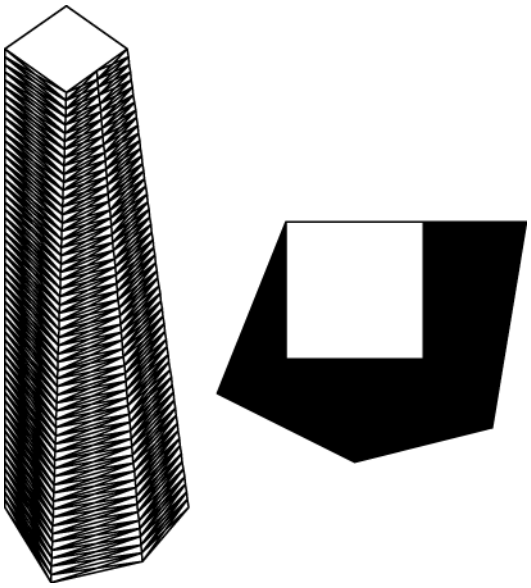

Figure 5.84a: PROG21a, no curve conversion and no rotation

The second example uses the same top and bottom points and a rotation of the morph of 180 degrees.

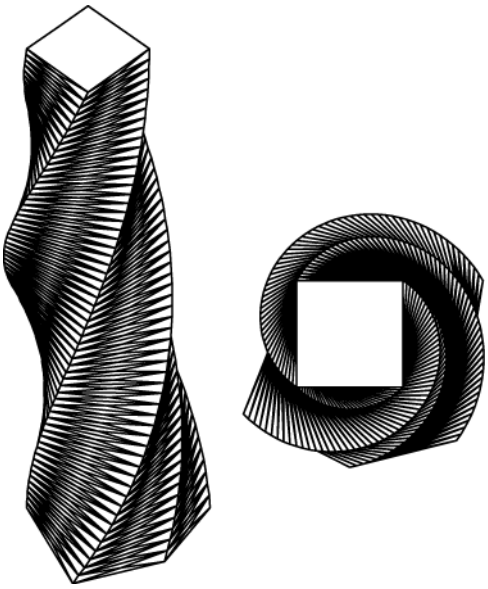

Figure 5.84b: PROG21a, no curve conversion with rotation

The third example uses the same top and bottom points, converted to arcs, curve fit, with no rotation.

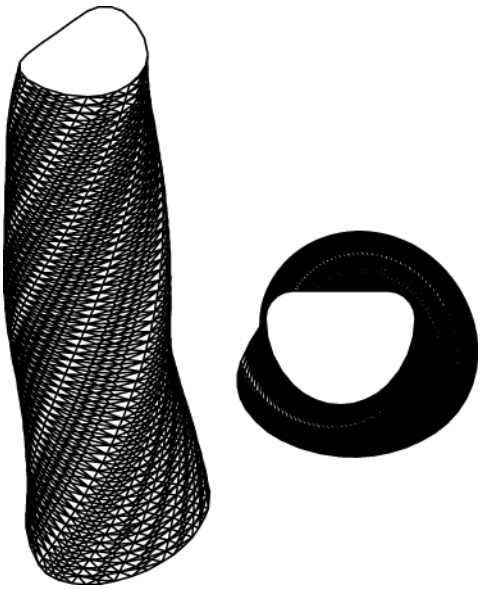

Figure 5.84c: PROG21a, curve fitted

The fourth example uses the same top and bottom points, converted to a Cubic B-spline, with no rotation.

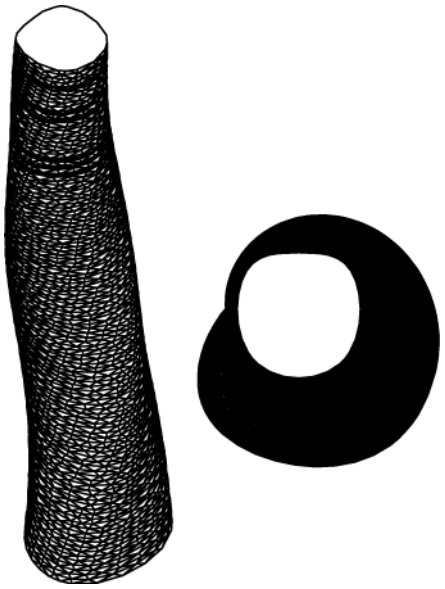

Figure 5.84d: PROG21a, Cubic B-spline

Sample script files for PROG21a:

```
;-----------------------------------
(prog21a)
; Layer name:
LAYER21a_1a
; number of levels
60
; level height
10'
; bottom list
((-80.0 -60.0) (0.0 -100.0) (80.0 -80.0)
(100.0 40.0) (-40.0 40.0) (-80.0 -60.0))
; top list
((-40.0 -40.0) (0.0 -40.0) (40.0 -40.0) (40.0
40.0) (-40.0 40.0) (-40.0 -40.0))
; rotation
0
; straight, fit, spline6, spline5
; section type
straight
;-----------------------------------
```

Add function PROG21a using a copy of PROG13. Modify the creation of the 3DMESH to a LOFTed solid as in PROG04a.

Add input for the type of curve conversion:

```
; straight, fit, spline6, spline5
(setq stype
 (strcase (getstring
 "\nEnter section type:")))
```

Convert the polyline from straight lines to a curve with:

```
; close polyline
(command "c")
; select polyline
(setq obj1 (ssadd (entlast)))
(command ".ZOOM" "e")
; convert polyline to a curve
; or spline
(if (/= stype "STRAIGHT") (progn
 (if (= stype "FIT") (progn
  (command ".PEDIT" obj1 "f" "")))
 (if (= stype "SPLINE6") (progn
  (setvar "SPLINETYPE" 6)
  (command ".PEDIT" obj1 "s" "")))
 (if (= stype "SPLINE5") (progn
  (setvar "SPLINETYPE" 5)
  (command ".PEDIT" obj1 "s" "")))
))
; make region
(command ".REGION" "last" "")
```

The computation of area and perimeter has also been added.

Completed function PROG21a:

```
;-----------------------------------
(defun prog21a ()
 (graphscr)
 ; tower repeats
 ; morph from lists of points
 (setvar "CMDECHO" 0)
 ; turn off undos
 (command ".UNDO" "c" "n")
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get curve parameters
 (princ "\nProg21a - Tower")
 (setq TowerLayer
   (getstring
   "\nEnter layer name: "))
 (setq NumLevels
   (getint
   "\nEnter number of levels: "))
 (setq LevelHeight
   (getdist
   "\nEnter level height: "))
 (setq BotSect
   (read (getstring
   "\nEnter bottom list of
   points: ")))
 (setq TopSect
   (read (getstring
   "\nEnter top list of points: ")))
 (setq RotTotal
   (getreal
   "\nEnter total rotation
   angle: "))
 ; straight, fit, spline6, spline5
 (setq stype
  (strcase (getstring
  "\nEnter section type:")))
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; set elev
 (setq LevelElev 0.0)
 ; area and perm
 (setq tarea 0.0)
 (setq tperm 0.0)
 ; points
 (setq npts (length BotSect))
 ; rotation inc
 (setq RotInc
   (/ RotTotal NumLevels))
 (setq Rotang 0.0)
 ; start LOFT selection list
 (setq plist (ssadd))
 ; create each level
 (setq nlevel 0)
 (repeat (+ NumLevels 1)
  ; start polyline
  (command ".PLINE")
  (setq npt 0)
  (repeat (- npts 1)
   ; get point
   (setq xpt
     (nth 0 (nth npt BotSect)))
   (setq ypt
     (nth 1 (nth npt BotSect)))
   ; compute inc
   (setq xinc
     (/ (- (nth 0 (nth npt TopSect))
```

```
    (nth 0 (nth npt BotSect)))
    NumLevels))
  (setq yinc
    (/ (- (nth 1 (nth npt TopSect))
    (nth 1 (nth npt BotSect)))
    NumLevels))
  ; add to pt
  (setq xpt (+ (* xpt 12)
    (* (* xinc 12) nlevel)))
  (setq ypt (+ (* ypt 12)
    (* (* yinc 12) nlevel)))
  ; rotate
  (setq rxpt
    (- (* xpt (cos (dtr Rotang)))
    (* ypt (sin (dtr Rotang)))))
  (setq rypt
    (+ (* xpt (sin (dtr Rotang)))
    (* ypt (cos (dtr Rotang)))))
  (setq npnt
    (list rxpt rypt LevelElev))
  ; add to polyline
  (command npnt)
  ; next pt
  (setq npt (+ npt 1))
 )
 ; close polyline
 (command "c")
 ; select polyline
 (setq obj1 (ssadd (entlast)))
 (command ".ZOOM" "e")
 ; convert polyline to a curve
 ; or spline
 (if (/= stype "STRAIGHT") (progn
  (if (= stype "FIT") (progn
   (command ".PEDIT" obj1 "f" "")))
  (if (= stype "SPLINE6") (progn
   (setvar "SPLINETYPE" 6)
   (command ".PEDIT" obj1 "s" "")))
  (if (= stype "SPLINE5") (progn
   (setvar "SPLINETYPE" 5)
   (command ".PEDIT" obj1 "s" "")))
 ))
 ; make region
 (command ".REGION" "last" "")
 ; area
 (command ".AREA" "o" "last")
 (setq tarea
  (+ tarea (getvar "AREA")))
 (setq tperm
  (+ tperm (getvar "PERIMETER")))
 ; add to LOFT list
 (setq plist
  (ssadd (entlast) plist))
 ; inc elev
 (setq LevelElev
   (+ LevelElev LevelHeight))
 ; inc level
 (setq nlevel (+ nlevel 1))
 ; inc rotation
 (setq Rotang (+ Rotang Rotinc))
)
(command ".ZOOM" "e")
; turn off history
(setvar "SOLIDHIST" 0)
; set DELOBJ to delete the sections
(setvar "DELOBJ" 1)
; set LOFTNORMALS =1 for smooth
; =0 for ruled
(setvar "LOFTNORMALS" 0)
; set LOFTPARAM to default
(setvar "LOFTPARAM" 7)
; loft sections
(command ".LOFT" plist "" "")
; turn on history
(setvar "SOLIDHIST" 1)
(setvar "CMDECHO" 1)
(command ".VIEW" "swiso")
(command ".ZOOM" "e")
(command ".HIDE")
; turn on undo
(command ".UNDO" "a")
; area and perm
(princ "\nLevels: ")
 (princ NumLevels)
  (princ "   Area: ")
  (princ (rtos (/ tarea 144) 2 0))
  (princ " sqft   Perm: ")
  (princ (rtos (/ tperm 12) 2 0))
  (princ " ft")
 (princ)
)
;-----------------------------------
```

A spline version of the floor section can also be created directly, without conversion from a polyline, using the polyline points as spline control points.

Here is a series of drawings showing a polyline created from lines, a polyline curve fitted with arcs, converted to a Cubic B-spline, spline type 6, and converted to a Quadratic B-spline, spline type 6. The final drawing creates the spline directly from the computed polyline points. The SPLINE command is used to create this curve.

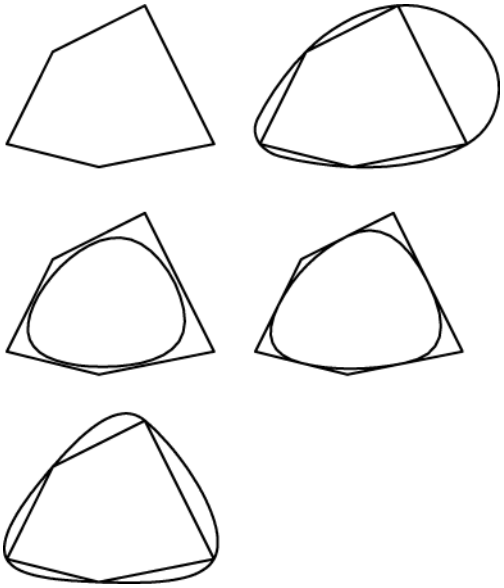

Figure 5.85a: Comparison of curve conversions

As noted before the points selected from which to create a spline can greatly change the curvature of the spline. Consideradding points at midpoints of edges and also ones very close to existing vertices.

Figure 5.85b: Splines with additional control points

In this example the previously used top and bottom points are morphed without rotation directly to splines.

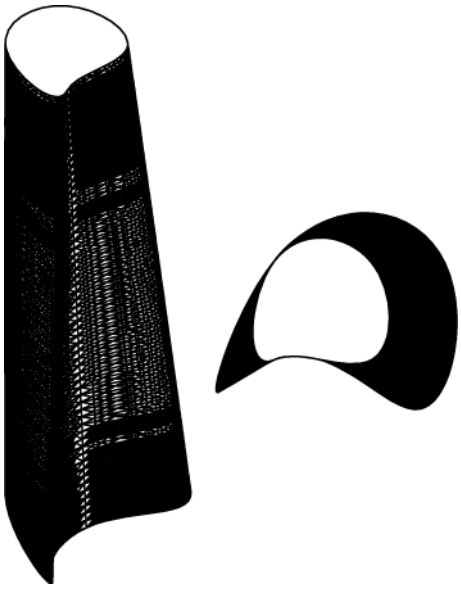

Figure 5.85c: PROG22a, spline floor plan

Sample script files for PROG22a:

```
;-----------------------------------
(prog22a)
; Layer name:
LAYER22a_1a
; number of levels
60
; level height
10'
; bottom list
((-80.0 -60.0) (0.0 -40.0) (80.0 -80.0)
(100.0 40.0) (-40.0 40.0) (-80.0 -60.0))
; top list
((-40.0 -40.0) (0.0 -40.0) (40.0 -40.0) (40.0
40.0) (-40.0 40.0) (-40.0 -40.0))
; rotation
0
;-----------------------------------
```

Results for this example:

```
Levels: 60
Area: 887364 sqft   Perm: 29350 ft
```

Add function PROG22a using a copy of PROG21a. Change the creation of the polyline for each floor to the creation of a spline.

Remove the input for type of curve conversion:

```
; straight, fit, spline6, spline5
 (setq stype
  (strcase (getstring
  "\nEnter section type:")))
```

Change the polyline to a spline, from:

```
; start polyline
(command ".PLINE")
```

To:

```
; start spline
(command ".SPLINE")
```

And change:

```
; add to polyline
(command npnt)
```

To:

```
; add to spline
(command npnt)
```

Change the conversion of the polyline to the close of the spline, from:

```
; close polyline
(command "c")
; select polyline
(setq obj1 (ssadd (entlast)))
(command ".ZOOM" "e")
; convert polyline to a curve
; or spline
(if (/= stype "STRAIGHT") (progn
 (if (= stype "FIT") (progn
  (command ".PEDIT" obj1 "f" "")))
 (if (= stype "SPLINE6") (progn
  (setvar "SPLINETYPE" 6)
  (command ".PEDIT" obj1 "s" "")))
 (if (= stype "SPLINE5") (progn
  (setvar "SPLINETYPE" 5)
  (command ".PEDIT" obj1 "s" "")))
))
```

To simply:

```
; close spline
(command "c" "")
(command ".ZOOM" "e")
```

Convert the tower solid model to a solid members model

As demonstrated in the Lineal and Circular Path models in the previous sections, the points being computed for each floor in the tower can also be used to develop vertical tubular members.

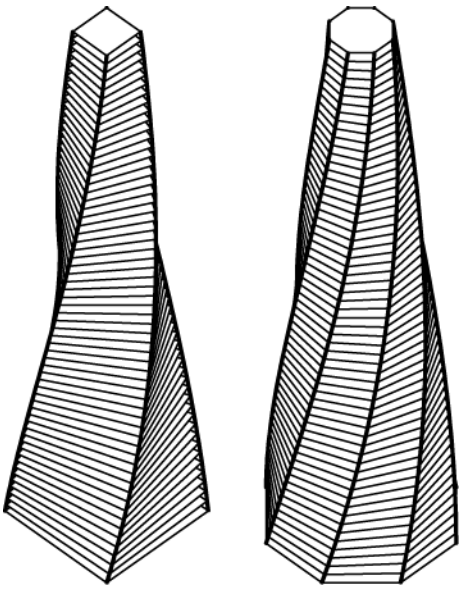

Figure 5.86a: PROG23a, tower model with tubular vertical members, four and eight sided polygons, rotated

In this example the vertices of the polygon are used to create a floor plate and are collected in a list so they can be vertically connected.

Add function PROG023a using a copy of PROG04a for the tower generation and PROG27 from the Lineal Path section for the sweeping of the vertical members.

Input for the section dimension has been added:

```
(setq sectrad1
  (getdist cpnt
  "\nEnter vertical member radius:"))
```

The selection list changes to a points list, from:

```
; start LOFT selection list
(setq plist (ssadd))
```

To:

```
; start points list
(setq flist (list ))
```

The number of points for the polygon changes, we do not want to duplicate vertices for the vertical members and add a points list, from:

```
; start polyline
(command ".PLINE")
(repeat (+ NumSides 1)
```

To:

```
  ; start polyline
  (command ".PLINE")
  (setq plist (list ))
  (repeat NumSides
```

When the polygon points are given, they are also saved in a points list, change from:

```
; add to polyline
(command npnt)
```

To:

```
; add to polyline
(command npnt)
; add to list
(setq plist
 (append plist (list npnt)))
```

Change the LOFT selection list to a floor points list, from:

```
; add to LOFT list
(setq plist
 (ssadd (entlast) plist))
```

To:

```
; make floor
(command ".REGION" "last" "")
; add to points list
(setq flist
 (append (list plist) flist))
```

The LOFT series of commands are removed at the end of the function and the SWEEP section from PROG27. Review function PROG23a for the changes in this section of the function.

Sample script files for PROG23a:

```
;-----------------------------------
(prog23a)
; Layer name:
LAYER23a_1_4
; number of levels
60
; level height
10'
; number polygon sides
4
; radius start
90'
; radius end
30'
; rotation
90
; vertical member radius
12
;-----------------------------------
```

Completed function PROG23a:

```
;-----------------------------------
(defun prog23a ()
 (graphscr)
 ; tower by polygon repeats
 ; twist tubular
 (setvar "CMDECHO" 0)
 ; center point
 (setq cpnt (list 0.0 0.0 0.0))
 ; get curve parameters
 (princ "\nProg23a - Tower")
 (setq TowerLayer
   (getstring
   "\nEnter layer name: "))
 (setq NumLevels
   (getint
   "\nEnter number of levels: "))
 (setq LevelHeight
   (getdist
   "\nEnter level height: "))
 (setq NumSides
   (getint
   "\nEnter number of polygon
   sides: "))
 (setq RadStart
   (getdist
   "\nEnter start radius: "))
 (setq RadEnd
   (getdist
   "\nEnter end radius: "))
 (setq RotTotal
   (getreal
   "\nEnter total rotation
   angle: "))
 (setq sectrad1
  (getdist cpnt
  "\nEnter vertical member radius:"))
 ; clear layer
 (command ".LAYER" "THAW" "*"
   "ON" "*" "")
 (command ".LAYER"
   "MAKE" TowerLayer "")
 (command ".LAYER" "SET" 0 "")
 (command ".LAYER" "OFF" "@*"
   "FREEZE" "@*" "")
 (command ".LAYER" "THAW" TowerLayer
   "ON" TowerLayer
   "SET" TowerLayer "")
 (command ".ERASE" "ALL" "")
 ; set elev
 (setq LevelElev 0.0)
 ; start points list
 (setq flist (list ))
 ; radius inc
 (setq RadInc
   (/ (- RadEnd RadStart)
   NumLevels))
 (setq Rad RadStart)
 ; rotation inc
 (setq RotInc
   (/ RotTotal NumLevels))
 (setq Rotang 0.0)
 ; create each level
 (repeat (+ NumLevels 1)
  ; draw polygon
  (setq panginc (/ 360.0 NumSides))
  (setq pang (/ panginc 2.0))
  ; start polyline
  (command ".PLINE")
  (setq plist (list ))
  (repeat NumSides
   ; compute point
   (setq xpt (+ (nth 0 cpnt)
     (* Rad (sin (dtr pang)))))
   (setq ypt (+ (nth 1 cpnt)
     (* Rad (cos (dtr pang)))))
   ; rotate point
   (setq rxpt
     (- (* xpt (cos (dtr Rotang)))
     (* ypt (sin (dtr Rotang)))))
```

```
   (setq rypt
     (+ (* xpt (sin (dtr Rotang)))
     (* ypt (cos (dtr Rotang)))))
   (setq npnt
     (list rxpt rypt LevelElev))
   ; add to polyline
   (command npnt)
   ; add to list
   (setq plist
    (append plist (list npnt)))
   ; inc ang
   (setq pang (+ pang panginc))
  )
  ; close polyline
  (command "c")
  (command ".ZOOM" "e")
  ; make floor
  (command ".REGION" "last" "")
  ; add to points list
  (setq flist
   (append (list plist) flist))
  ; inc radius
  (setq Rad (+ Rad RadInc))
  ; inc elev
  (setq LevelElev
    (+ LevelElev LevelHeight))
  ; inc rotation
  (setq Rotang (+ Rotang Rotinc))
 )
 (command ".ZOOM" "e")
 ; vertical members
 (setq ipt 0)
 (repeat NumSides
  (setq ifloor 0)
  ; start 3DPOLY
  (command ".3DPOLY")
  (repeat (+ NumLevels 1)
    ; add pt to 3DPOLY
    (setq npt
     (nth ipt (nth ifloor flist)))
    (command npt)
    ; next frame
    (setq ifloor (+ ifloor 1))
  )
  ; close
  (command "")
  (setq obj1 (ssadd (entlast)))
  ; add cross section
  (command ".CIRCLE" npt sectrad1)
  (setq obj2 (ssadd (entlast)))
  (command ".ZOOM" "e")
  ; SWEEP members across frames
  ; turn off history
  (setvar "SOLIDHIST" 0)
  ; set DELOBJ to delete the sections
  ; and path
  (setvar "DELOBJ" 2)
  ; sweep sections
  (command ".SWEEP" obj2 "" obj1)
  ; turn on history
  (setvar "SOLIDHIST" 1)
  (command ".ZOOM" "e")
  ; next pt
  (setq ipt (+ ipt 1))
 )
 (princ)
)
;-----------------------------------
```

With a small amount of reorganization of function PROG04a and PROG23a, three elements of the tower can be generated; the skin, the floor plates, and the primary vertical members.

The location of the vertical members is based on computed points. If the floor plate was a spline, as in function PROG22a, the points would have to computed using the DIVIDE command.

Another version of this function could include instead of the floor plate a horizontal member at edge of the floor using sweeping the generated polygon. This could occur at each floor or some entered increment, every 5 floor.

Additional vertical members can also be considered if the sides of the polygon are subdivided.

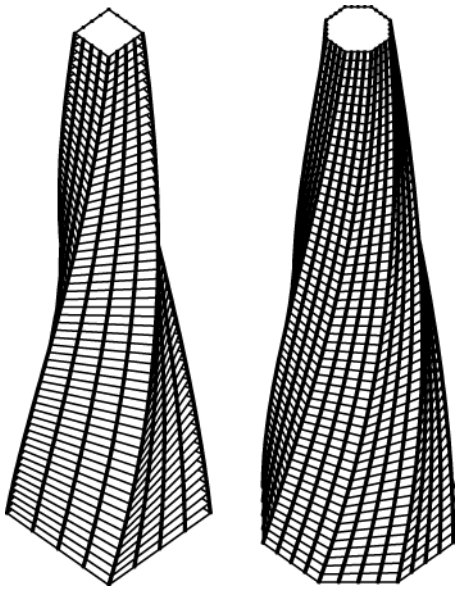

Figure 5.86b: PROG23b, tower model with additional tubular vertical members, four and eight sided polygons, rotated

Sample script file for PROG23b:

```
;-----------------------------------
(prog23b)
; Layer name:
LAYER23b_1_4
; number of levels
60
; level height
10'
; number polygon sides
4
; radius start
90'
; radius end
30'
; rotation
90
; vertical member radius
12"
; midpts
3
;-----------------------------------
```

Add function PROG23b using a copy of PROG23a. Once the polygon points are computed, find subdivision points along each edge.

Add input for the number of subdivision to compute along each edge:

```
(setq midpts
  (getint
  "\nEnter number midpts:"))
```

Following the REGION command:

```
; make floor
(command ".REGION" "last" "")
```

Add:

```
; midpt factor and pt list
(if (> midpts 0) (progn
```

```
(setq dfact
 (/ 1.0 (+ midpts 1)))
(setq mlist
 (append plist (list (nth 0 plist))))
; add midpts
(setq ipt 0)
(repeat NumSides
 (setq pt1 (nth ipt mlist))
 (setq pt2 (nth (+ ipt 1) mlist))
 (setq mfact dfact)
 (repeat midpts
  (setq mpnt
   (polar pt1 (angle pt1 pt2)
    (* (distance pt1 pt2) mfact)))
   ; add to list
   (setq plist
    (append plist (list mpnt)))
   ; inc midpt
   (setq mfact (+ mfact dfact))
   )
 (setq ipt (+ ipt 1))
 )
))
```

A midpoint factor is computed, the first point of the polygon is appended to a temporary point list for these computations. Every two points are extracted from the list, the distance and the angle are found between them, and the midpoint is computed and appended to the floor plate points list.

The generation of the vertical members is change from:

```
; vertical members
(setq ipt 0)
(repeat NumSides
```

To account for the extra points computed:

```
; vertical members
(setq ipt 0)
(repeat
 (+ NumSides (* NumSides midpts))
```

Another consideration could be a separate section radius for these additional members.