Appendix D. Scheme Example Files

Topic: Deformable Surfaces

ADM provides three example Scheme files, which can aid in better understanding the deformable service routines. The files are called dsmouse.scm, dsdemo.scm, and dscurv.scm. Once these Scheme files have been loaded into a Scheme application, such as Scheme AIDE, other Scheme methods are available to experiment with the deformable surfaces and curves.

dsmouse.scm

Topic: Deformable Surfaces

Contains:

```
ds:mouse (face)
ds:mouse–help
ds:mouse–view–help
ds:select–face
ds:set–mouse–slider (target_tag gain_range)
ds:get–mouse–slider
```

Purpose . . . . Provides the rubberband driver to allow deformable curve and surface sculpting with the mouse. Also provides the convenience routines, (ds:select–face) and (ds:select–edge) used to select faces from solid modeling for sculpting.

Rubberband overview	The mouse is bound to a particular face or edge with the command (ds:mouse entity). When executed it outputs the following help table to act as a reminder of the mouse functions. Whenever the cursor is placed into a viewing window the entity bound to the mouse is examined to see if one of its tag identities is highlighted. The side effect of this check is to build a deformable modeling attribute for an entity which does not have one. As a convenience one can execute the function (ds:mouse '( )) which will preserve the mouse view manipulation capabilities but will disable the deformable surface functions. Now when the cursor is placed in a rendering window no side effects will occur. When (ds:mouse '( )) is executed the shorter function table message shown below is printed in the dialog box.

```
Note: Bind a load to the slider function with

    (ds:set-mouse-slider tag gain_inc)

Note: Release select keys after mouse moves begin
```

	Left Button	Right Button	Left and Right Buttons
Plain	RotView	ZoomView	PanView
Shift	Track Control Point Z	Add/Toggle Control Point	
Shift Drag		Add Patch	
Control	Track Control Point XY	Add Point Pres-sure	
Shift Control	Slide Gain	Remove Tag	

Lft Column	Actions taken when left mouse key is pressed
Rgt Column	Actions taken when right mouse key is pressed.
Rgt_Lft Column	Actions taken when left and right mouse keys are pressed.
Plain Row	Actions taken when mouse keys are pressed alone.

Shift Row	Actions taken when mouse and shift keys are pressed.
Shft–drag Row	Actions taken when mouse and shift keys are pressed and dragged (i.e. moved).
Ctrl Row	Actions taken when mouse and Ctrl keys are pressed.
Shift_Ctrl Row	Actions taken when mouse, shift, and Ctrl keys are pressed.
Cpt	Constraint Point.
Ppt	Pressure Pt Load.
RotView	Virtual ball manipulation of the view.
ZoomView	Track cursor.y to animate view focal distance.
PanView	Track cursor to animate view focal point.
Track_Cpt_Z	Move a Cpt parallel to the *z*-axis.
Track_Cpt_XY	Move a Cpt parallel to the *xy*-plane.
Slide_gain	Track cursor.x to animate one load's gain, refer to (ds:set–mouse–slider target_tag gain_range).
Add/Tog_Cpt	When a constraint is highlighted, toggle its enabled/disabled state, else add a Cpt at intersection of pick–ray and surface.
Add_Ppt	Add Ppt at intersection of pick–ray and surface.
Rm_Tag	Remove highlighted Cpt or Ppt.

ds:mouse

Topic: Deformable Surfaces

Action: Binds or clears the binding of a face or edge to the mouse driven deformable modeling rubberband driver.

APIs: None

Syntax: (**ds:mouse** entity [solve_flag])

Arg Types:	
entity	face | edge
solve_flag	integer

Returns: unspecified

Errors: None

Description: Binds entity to the deformable modeling rubberband driver. Once bound the mouse in combination with the shift and control keys can be used to add, modify, and remove constraints, pressure loads, and patches. It can be used to track constraint points in the *z* direction or in the *xy* plane. The mouse can be used to slide the gain of one specified gain (refer to ds:set–mouse–slider). Additionally the mouse can modify the viewing angle.

Calling ds:mouse with a null argument will bind only the view manipulation commands to the mouse and clear the deformable modeling functions.

The mouse functionality includes:

move–left–button	Rotate view
move–right–button	Zoom view
move–left–right–button	Track view
move–right–button	Zoom view
move–left–right–button	Track view
move–shift–left–button	Track constraint point in *z*-direction
shift–right–button	When cursor points to a constraint, toggle constraint enable/disable state; else add constraint point at cursor position
shift–right–drag–button	Add a square patch to a parent patch whose opposite corners are defined by the mouse–down and mouse–up events
ctrl–right–button	Add pressure point at cursor position
move–shift–ctrl–left–button	Track a load gain
shift–ctrl–right–button	Remove tag–object at cursor position

Limitations: Not Applicable

Example:

```
; ds:mouse
; If not already loaded, then
(load ”dsdemo.scm”)
;; [Autoloading dscurve.scm]
;; [Autoloading demoinit.scm]
; define some helpful globals
(define ds–model)
; build a test square spline face
; (6x6 control points, x and y side length = 36)
(set! ds–model (ds:test–face 6 6 36 36 0))
(entity:erase ds–model)
; don’t render the face
; bind this face to the mouse
(ds:mouse ds–model)
;; ds:mouse Function Map
;; Note: Bind a load to the slider function with
;; (ds:set–mouse–slider tag gain_inc)
;; Note: Release select keys after mouse moves begin
```

Note: Bind a load to the slider function with

```
(ds:set–mouse–slider tag gain_inc)
```

Note: Release select keys after mouse moves begin

	Left Button	Right Button	Left and Right Buttons
Plain	RotView	ZoomView	PanView
Shift	Track Control Point Z	Add/Toggle Control Point	
Shift Drag		Add Patch	
Control	Track Control Point XY	Add Point Pres-sure	
Shift Control	Slide Gain	Remove Tag	

```
; clear the deformable bindings –
; save the view manipulations
(ds:mouse ’())
;; ds:mouse Function Map
```

Left Button	Right Button	Left and Right Buttons
RotView	ZoomView	PanView

ds:mouse–help

Topic: Deformable Surfaces

Action: Displays the mouse functions established by the command ds:mouse.

APIs: None

Syntax: (**ds:mouse–help**)

Arg Types: None

Returns: unspecified

Errors: None

Description: Displays the mouse functions established by the command (ds:mouse entity).

Limitations: Not Applicable

Example:

```
; ds:mouse–help
; If not already loaded, then
(load ”dsdemo.scm”)
;; [Autoloading dscurve.scm]
;; [Autoloading demoinit.scm]
; displays the mouse help
(ds:mouse–help)
;; ds:mouse Function Map
;; Note: Bind a load to the slider function with
;; (ds:set–mouse–slider tag gain_inc)
;; Note: Release select keys after mouse moves begin
```

Note: Bind a load to the slider function with

```
(ds:set-mouse-slider tag gain_inc)
```

Note: Release select keys after mouse moves begin

	Left Button	Right Button	Left and Right Buttons
Plain	RotView	ZoomView	PanView
Shift	Track Control Point Z	Add/Toggle Control Point	
Shift Drag		Add Patch	
Control	Track Control Point XY	Add Point Pres-sure	
Shift Control	Slide Gain	Remove Tag	

ds:mouse–view–help

Topic:	Deformable Surfaces
Action:	Displays the deformable model rubber–band driver functions supported by the command (ds:mouse ’()).
APIs:	None
Syntax:	(**ds:mouse-view-help**)
Arg Types:	None
Returns:	unspecified
Errors:	None
Description:	Displays the deformable model rubber–band driver functions supported by the command (ds:mouse ’( )).

The mouse functionality includes:

move–left–button	rotate view.
move–right–button	zoom view.
move–left–right–button	track view.

Limitations: Not Applicable

Example:
```
; ds:mouse-view-help
; If not already loaded, then
(load ”dsdemo.scm”)
;; [Autoloading dscurve.scm]
;; [Autoloading demoinit.scm]
(ds:mouse-view-help ds-face)
;; ds:mouse Function Map
```

Left Button	Right Button	Left and Right Buttons
RotView	ZoomView	PanView

ds:select–face

Topic: Deformable Surfaces

Action: Selects a face from a solid model and prepares it for deformable modeling.

APIs: None

Syntax: (**ds:select-face**)

Arg Types: None

Returns: unspecified

Errors: None

Description: Prompts for a pick–event, which is used to select a face from a solid model for deformable editing. When the pick–event successfully selects a face, that face is bound to the mouse with (ds:mouse selected–face), and is bound to the global variable ds–model so that all the convenience routines provided in the file dsdemo.scm will work on the selected–face.

Limitations: Not Applicable

Example:

```
; ds:select-face
; If not already loaded, then
(load ”dsdemo.scm”)
;; [Autoloading dscurve.scm]
;; [Autoloading demoinit.scm]
; Select one face out of a solid to sculpt
(ds:select-face)
; Pick Face to Sculpt:
; the end-user places the cursor over a
; face and picks
; with the left mouse.
;; #[entity 5 1] constructed new attrib_dm2acis
;; now bound to ds-face
;; ds:mouse Function Map
;; Note: Bind a load to the slider function with
;; (ds:set-mouse-slider tag gain_inc)
;; Note: Release select keys after mouse moves begin
```

Note: Bind a load to the slider function with

```
(ds:set-mouse-slider tag gain_inc)
```

Note: Release select keys after mouse moves begin

	Left Button	Right Button	Left and Right Buttons
Plain	RotView	ZoomView	PanView
Shift	Track Control Point Z	Add/Toggle Control Point	
Shift Drag		Add Patch	
Control	Track Control Point XY	Add Point Pres- sure	
Shift Control	Slide Gain	Remove Tag	

ds:select–edge

Topic: Deformable Surfaces

Action: Selects an edge from a wire model and prepares it for deformable modeling.

APIs: None

Syntax: (**ds:select-edge**)

Arg Types: None

Returns: unspecified

Errors: None

Description: Prompts for a pick–event, which is used to select an edge from a wire model for deformable editing. When the pick–event successfully selects an edge, that edge is bound to the mouse with (ds:mouse selected–edge), and is bound to the global variable ds–model so that all the convenience routines provided in the file dsdemo.scm will work on the selected–edge.

Limitations: Not Applicable

Example:

```
; ds:select–edge
; If not already loaded, then
(load ”dsdemo.scm”)
;; [Autoloading dscurve.scm]
;; [Autoloading demoinit.scm]
; Select one edge out of a wire to sculpt
(ds:select–edge)
; Pick EDGE to Sculpt:
; the end–user places the cursor over an
; edge and picks with the left mouse.
;; #[entity 5 1] constructed new attrib_dm2acis
;; now bound to ds–model
;; ds:mouse Function Map
;; Note: Bind a load to the slider function with
;; (ds:set–mouse–slider tag gain_inc)
;; Note: Release select keys after mouse moves begin
```

Note: `Bind a load to the slider function with`

```
(ds:set-mouse-slider tag gain_inc)
```

Note: `Release select keys after mouse moves begin`

	Left Button	Right Button	Left and Right Buttons
Plain	RotView	ZoomView	PanView
Shift	Track Control Point Z	Add/Toggle Control Point	
Shift Drag		Add Patch	
Control	Track Control Point XY	Add Point Pressure	
Shift Control	Slide Gain	Remove Tag	

ds:get–mouse–slider

Topic: Deformable Surfaces

Action: Returns the current tag value and slider bar range in use for the mouse driven slider bar used by the deformable modeling mouse package.

APIs: None

Syntax: (**ds:set–mouse–slider** tag gain–range)

Arg Types:	
tag	integer
gain–range	real

Returns: unspecified

Errors: None

Description: Returns the current values bound to the global variables, data–ds:mouse–slider–tag and data–ds:mouse–slider–gain–inc. These two variables are used by the mouse slider bar function, move–shift–ctrl–left–button. Each time the mouse moves while the shift, ctrl, and left buttons are depressed the deformable rubberband driver computes a new gain value, assigns that value to the tag identified tag–object, resolves the deformable equations, and displays the result. Moving the mouse fully across the screen from left to right increases the gain value by an amount of gain–range. Moving from right to left decreases that value.

Limitations: Not Applicable

Example:

```
; ds:get-mouse-slider
; If not already loaded, then
(load ”dsdemo.scm”)
;; [Autoloading dscurve.scm]
;; [Autoloading demoinit.scm]
; define some helpful globals
(define ds-model)
; build a test square spline face
; (6x6 control points, x and y side length = 36)
(set! ds-model (ds:test-face 6 6 36 36 0))
(entity:erase ds-model)
; don’t render the face
; bind the face to the mouse
(ds:mouse ds-model)
; add a pressure load
(ds:add-dist-press ds-mouse 0)
;; (5)
; bind this load to the slider bar
(ds:set-mouse-slider 5 10000)
; hold the shift and ctrl keys down. press and hold
; the left mouse button down. Moving the cursor from
; left to right and back again will deform the
; surface in a continuous manner.
; check the current mouse slider bar values
(ds:get-mouse-slider)
;; (5 10000)
```

ds:set–mouse–slider

Topic: Deformable Surfaces

Action: Binds a tag load number to the mouse driven slider bar in preparation for animation like sculpting..

APIs: None

Syntax: (**ds:set-mouse-slider** tag gain-range)

Arg Types:	tag	integer
	gain–range	real

Returns: unspecified

Errors: None

Description: Binds values to the global variables, data–ds:mouse–slider–tag and data–ds:mouse–slider–gain–inc. These two variables are used by the mouse slider bar function, move–shift–ctrl–left–button. Each time the mouse moves while the shift, ctrl, and left buttons are depressed the deformable rubberband driver computes a new gain value, assigns that value to the tag identified tag–object, resolves the deformable equations, and displays the result.

gain value assigns that value to the tag identified tag–object, resolves the deformable equations, and displays the result. Moving the mouse fully across the screen from left to right will increase the gain value by an amount of gain–range. Moving from right to left will decrease that value.

Limitations: Not Applicable

Example:

```
; ds:set-mouse-slider
; If not already loaded, then
(load "dsdemo.scm")
;; [Autoloading dscurve.scm]
;; [Autoloading demoinit.scm]
; define some helpful globals
(define ds-model)
; build a test square spline face
; (6x6 control points, x and y side length = 36)
(set! ds-model (ds:test-face 6 6 36 36 0))
(entity:erase ds-model)
; don't render the face
; bind the face to the mouse
(ds:mouse ds-model)
; add a pressure load
(ds:add-dist-press ds-face 0)
;; (5)
; bind this load to the slider bar
(ds:set-mouse-slider 5 10000)
; hold the shift and ctrl keys down. press and hold
; the left mouse button down. Moving the cursor from
; left to right and back again will deform the
; surface in a continuous manner.
```

dsdemo.scm

Topic: Deformable Surfaces

The dsdemo.scm file provides a set of Scheme functions designed to help demonstrate the capabilities of deformable curve and surface sculpting. A group of canned demonstration sequences are included to show how the individual features of sculpting work. In particular, point constraints, curve constraints, curve_loads, and distributed pressures are highlighted. Two demos called goblet and space–bug are provided to show how a sequence of sculpting commands can be used to build interesting object shapes.

These demos are intended to be interactive. Running a demo command creates an interesting test case meant to be a starting point for experimenting with the deformable sculpting capabilities.

The easiest thing to do after running a canned–demo is to use the shift–left mouse key combination to track point constraints. As the point constraint is tracked you will be able to see how the different tag–objects affect the shape of the surface. But by no means stop there. Feel encouraged to modify, add, or remove constraints and loads to experiment with combinations of features.

To help simplify this experimentation, an additional family of convenience routines are provided which shorten the amount of typing needed to execute the ds:command Scheme extensions. These commands are for convenience and by their mere existence to act as subtle suggestions of things to try. Deformable surface sculpting is a general purpose editing function. The convenience commands and the ds:command extensions may be run in any combination and in any order.

Demo Functions

Topic: Deformable Surfaces

Use these functions to start cases and then experiment with the commands below.

(pinned–square . n)	build an *nxn* square (default *n=6*) with center and corner pt–cstrns. Try tracking the point constraints
(hinged–square . n)	build an *nxn* square (default *n=6*) constrained with a center constraint point and 4 edge curve constraints. Try tracking the center point constraint.
(hung–square . n)	build an *nxn* square (default *n=6*) with a center constraint point and 4 edge curve spring loads. Try tracking the center point constraint. Try changing the 4 curve spring stiffness with the command (hung–gain gain–value).
(gravity–square . n)	build an *nxn* control–pt square (default n=6) a center–pt cstrn and vector–load. Try tracking the vector load.

(attractor–square . n)	build an *nxn* control–pt square (default n=6) with centered pt–cstrn and attractor load. Try tracking the attractor load.
(tangent–square . n)	build an *nxn* control–pt square (default n=6) with center and corner position–tangent pt–cstrns. Try tracking the central tangent constraint
(folding–square . n)	build an *nxn* square (default *n=6*) using inhomogeneous material properties and default shape to show folding caused by a tracked point constraint.
(corner–00 . n)	build an *nxn* control point square (default *n=6*) with internal linear curve constraint. Track the point constraint.
(par–00 . n)	build an *nxn* control point square (default *n=6*) with parabolic curve constraint. Track the point constraint.
(circ1–cstrn . n)	build an *nxn* square (default *n=6*) with a centered circle curve constraint and a point constraint. Try tracking the point constraints
(circ2–cstrn . n)	build an *nxn* control point square (default *n=6*) with an off–centered circle curve constraint and a point constraint. Try tracking with dynamic effects (dyn .1 15 1).
(circ1–load . n)	build an *nxn* square (default *n=6*) with a centered circle curve load and a point constraint. Try tracking the point constraints. Try varying the load gain with (gain ds–cc1 new_gain)
(circ2–load . n)	build an *nxn* control point square (default *n=6*) with an off–centered circle load and a point constraint. Try tracking the point constraints. try varying the load gain with (gain ds–cc1 new_gain)
(hinged–circ . n)	build an *nxn* control point square (default *n=9*) with a centered circle curve load, a distributed pressure (ds–cc2), and 4 edge curve constraints. Try changing the pressure gain, e.g. (gain ds–cc2 2000).
(goblet)	A *10x7* control point square sculpted into a goblet with constraints and loads. The sequence has several pauses of length *t* in 1/1000 of seconds. *t* defaults to 0. Try varying load gains of tag–objects ds–dp1 and ds–dp2.
(space–bug . n m)	An nxm control–pt square with curve tangent constraints.

System Modeling Functions

Topic: Deformable Surfaces

do–nurbs( )	force dsdemo.scm to build NURB faces.
do–nubs( )	force dsdemo.scm to build NUB faces.
clear( )	sequence to clear all ds demo data.
ds–pick(. e)	binds a picked face to ds–face for further sculpting. e = element count in each directory (default 3).
solve(. n)	call solver n times (default 1)
end( )	end_sculpting without committing to face
commit( )	save dsurf data in ACIS face.
dface (n m)	start over with [nxm] control point face
shape (flag)	set_default_shape flag = 1, use current shape flag = 0, set default to zero

mesh (n . m flag)	set u and v elem counts
	m not needed for edges
	m defaults to n for faces
	flag=0: fit everywhere (default)
	flag=1: fit cstrns
	flag=2: use best fit

split (u v flag)	split surf with dpt[u v] flag=0: split elem center containing [uv] flag=1: split exactly at [uv]
inc–degree( )	increase the surface’s polynomial degree by one. Avoid going above degree 3 surfaces. Use split to add more dofs.
slider (tag inc)	set mouse slider to tag item with given gain_inc
talking	toggle amount of text to command line.

Rendering and Reports

Topic: Deformable Surfaces

a( )	(b = ’all’) render loads, constraints, seams, and constraints
b( )	(b = ’basic’) render loads and constraints

c( )	(c = “control points”) render NURB and NUB control points
e( )	(e = ’elements’) render loads, constraints, and elem boundaries
g( )	(g = ’gauss–points’) render loads, constraints, and gauss points
k( )	(k = ’curvature’) render loads, constraints, and curvature combs
t( )	(t = ’tangents’) render loads, constraints, and tangent constraints
s( )	(s = ’seams’) render loads, constraints, and seams
ek( )	render ’elems’ + ’curvature’
es( )	render ’elems’ + ’seams’
est( )	render ’elems’ + ’seams’
ks( )	render ’curvature’ + ’seams’
kt( )	render ’curvature’ + ’tangents’
kst( )	render ’curvature’ + ’seams’ + ’tangents’
st( )	render ’seams’ + ’curvature’
grid(n)	render grid set to *nxn*.
d( )	(d = ’debug’) print a text description of ds–model
set–comb(cnt,gain)	set curvature comb graphics, cnt= number of tines per elem, gain = length of curvature vectors
size(r)	set icon size radius = r
disp–gain(g . tgt	set graphic scaling for tang_vec cstrns.
tag–type(tag)	return tag_type identifier or –1 for invalid tag
tag–loc(tag)	return tag position or () when pos is undefined
pick–tag	pick tag object to get its tag value.

DSURF Model Parameters

Topic: Deformable Surfaces

alpha (au . av)	set resistance to stretch (au = av = val; atheta = 0)

beta (bu . bv)	set resistance to bending (bu = bv = val; btheta = 0)
gamma (val)	set resistance to bending change rate value (g = 10)
delta(val)	set resist motion from default(d1 = val)
dyn (dt, m, d)	set time integration parameters(m=mass, d=damp)

DSURF Edge Conditions

Topic: Deformable Surfaces

open (dir)	set surface end condition to open dir: 0=u or 1=v
closed (dir)	set surface end condition to closed dir: 0=u or 1=v
periodic (dir)	set surface end condition to periodic dir: 0=u or 1=v
singular (dir s)	set surface singularity, dir: 0=u or 1=v s:0=none,1=lo,2=hi,3=both
su0, su1, su2, su3	singular u none, low, high, both
sv0, sv1, sv2, sv3	singular v none, low, high, both

Remove, Toggle, Change Constraints and Loads

Topic: Deformable Surfaces

delete (tag)	delete cstrn or load (tag=an int identifier)
dp4( )	delete the points added by p4()
toggle (tag)	toggle cstrn on/off state(tag=an int identifier)
t4( )	toggle cstrns (1 2 3 4) (usually the bounding curve constraints)
suv (tag uv)	set a pt cstrn/load uv value(tag=an int identifier)
sxyz (tag pos)	set a pt cstrn xyz value(tag=an int identifier)
conv(tag)	convert crv_cstrn into a crv_load
behavior(tag str)	set cstrn’s behavior “position” “pos_tan” “tangent”

Add Constraints

Topic: Deformable Surfaces

pc (u v)	add pt constraint at u v loc
tc(u v)	add tang constraint at u v loc
ptc(u v)	add pos_tan pt constraint at u v loc.
p4( )	add pt_cstrns on 4 corners
cc (uv0 uv1)	add straight curve constraint from uv0 to uv1
str–cstrn (uv0 uv1)	is the same as cc(uv0 uv1)
cir–cstrn (u v r)	add circle cstrn with ctr=u,v rad = r
par–cstrn( )	apply a test parabola crv_cstrn

Add, Edit Loads

Topic: Deformable Surfaces

gain (tag gain_val)	set_load_gain for load with given tag
pl (uv . gain)	add pressure pt at uv loc with gain (default 100)
f5 ( )	add 5 pt_loads near middle
af (gain)	increment all pt load gains
sf (gain)	set all pt load gain values
spr ( )	add spring at uv=(0.5 0.5)
asf (gain)	increment all spring load gains
ssf (gain)	set all spring load gain values
dp (. gain u0 v0 u1 v1)	add distributed pressure with gain (default 500)
adp (gain)	add gain to distributed pressure
sdp (gain)	set distributed pressure gain value
gravity (. g)	add vector load.
attractor(. g)	add attractor load.

cl (u0 v0 u1 v1)	add str curve load from uv0 to uv1
clz (u z r . g)	add u–param str curve to circle load
cir–load(u v r. g)	apply a circular load u, v=ctr, r=rad, g=gauss_pt_accuracy
par–load ( )	apply a test parabola crv_load(tag=parab_load)

dscurv.scm

Topic: Deformable Surfaces

The dscurv.scm file provides a set of Scheme functions designed to help demonstrate the capabilities of deformable curve sculpting. A group of canned demonstration sequences are included to show how the individual features of sculpting work for deformable curves.

Curve Demo Functions

Topic: Deformable Surfaces

Use these functions to start cases and then experiment with the commands below.

(flight–path)	Fit a deformable curve to some data. Options include, control–point count (def=30), u_min (def=0), u_max (def=100), samp_count (def=121), inc_cnt (def=3).
(pull–path)	flight–path with default shape ready for constraint point pulling. Shows how a shape can be modified from a given starting point.
(pinned–edge . n)	Build an n control–point edge (default n=9) with center and corner point–constraints. Try tracking the point–constraints.
(hinged–edge . n)	Build an n control–point edge (default n=9) using a tangent constraint at its center. Try tracking point and tangent constraints.
(pinned–hinged–edge . n)	Build an n control–point edge (default=9) using a tangent and position constraint at its center. Try tracking point and tangent constraints.
(pinned2–edge . n)	Build an n control–point edge (default n=9) with 2 interior and 2 end point constraints. Try tracking the point–constraints.

(hinged2–edge . n)	Build an n control point edge (default n=9) with 2 interior tangent and 2 end point constraints. Try tracking the point and tangent constraints.
(pinned–hinged2–edge . n)	Build an n control point edge (default n=9) using 2 interior tangent/position and 2 end position constraints. Try tracking point and tangent constraints.
(folding–edge . n)	Build an n control–point edge (default n=9) using default shape to show folding caused by a tracked point–constraint.
ecm0	no tag objects.
ecm1	added pt
ecm2	added distributed pressures
ecm3	added spring load
ecm4	added curve load
ecm5	added multiples of above
ecm6	added deletes and multiple commits