Chapter 2.
Scheme Extensions

Topic: Ignore

Scheme is a public domain programming language, based on the LISP language, that uses an interpreter to run commands. ACIS provides extensions (written in C++) to the native Scheme language that can be used by an application to interact with ACIS through its Scheme Interpreter. The C++ source files for ACIS Scheme extensions are provided with the product. *Spatial*'s Scheme based demonstration application, Scheme ACIS Interface Driver Extension (Scheme AIDE), also uses these Scheme extensions and the Scheme Interpreter. Refer to the *3D ACIS Online Help User's Guide* for a description of the fields in the reference template.

lop:convert–pipes

Scheme Extension: Local Operations

Action: Replaces the geometry of any faces in the supplied array which currently are pipe_spl_sur splines with rb_blend_spl_sur splines.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_convert_pipes

Syntax: `(lop:convert-pipes body-or-face-list [acis-opts])`

Arg Types:

body–or–face–list	body | face | (face ...)
acis–opts	acis–options

Returns: body

Errors: Refer to the Errors listed for the Scheme extension, lop:tweak–faces.

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: Replaces the geometry of any faces in the supplied array which currently are pipe_spl_sur splines with rb_blend_spl_sur splines. If a body is given, all faces in the body are converted.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

body–or–face–list identifies faces of a body to be moved.

acis–opts specifies options such as versioning and journaling.

Limitations: Refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Example:

```
; lop:convert–pipes
; Create a cylinder
(define cyl1
    (solid:cylinder (position 0 0 –25)
    (position 0 0 25) 25))
;; cyl1
; Create a block
(define block1
    (solid:block (position –50 –25 –25)
    (position 0 25 25)))
;; block1
; Subtract bodies
(define subtract (bool:subtract cyl1 block1))
;; subtract
; Taper top face of body
(define taper (lop:taper–faces (pick:face
    (ray (position 1 0 0) (gvector 0 0 1)))
    (position 0 0 0) (gvector 1 0 0) 45))
;; taper
; Blend top edge
(define blend (solid:blend–edges (pick:edge
    (ray (position 25 0 0) (gvector 0 0 1))) 10))
;; blend
; Convert splines
(define convert (lop:convert–pipes
    (pick:face (ray (position 24 0 0)
    (gvector 0 0 1)))))
;; convert
```

lop:edge–taper–faces

Scheme Extension: Local Operations

Action: Tapers an array of faces about an array of corresponding edges and a supplied draft vector by a given draft angle.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_edge_taper_faces

Syntax:
```
(lop:edge–taper–faces face–list edge–list direction
    angle [box–h box–l] [acis–opts])
```

Arg Types:

face–list	face | (face ...)
edge–list	edge | (edge ...)
direction	gvector
angle	real
box–h	position
box–l	position
acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the errors listed for the Scheme extension, lop:tweak–faces.

Some of the following errors result in an entity being highlighted where the error occurs. The entity type follows the error message below.

Valid direction must be supplied, or error;
LOP_TAP_BAD_NORMAL
Valid angle (between –90 and +90 degrees, and non zero) must be supplied or error;
LOP_TAP_BAD_ANGLE
Surface must be able to be tapered as requested, or error;
LOP_TAP_NO_SURF
Supplied edges must lie on corresponding faces, or error;
LOP_TWK_NO_EDGE
Supplied edges must not be parallel to draft vector at any point, or error;
LOP_TWK_NO_EDGE

Normal of face to be tapered should not be parallel to draft vector, or error;
LOP_TAP_NO_SURF (FACE*)

Must be able to find a face adjacent to vent face if a vent face is inserted, or error;
LOP_TAP_NO_ADJ_FACE

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: The purpose of edge taper–faces is to facilitate extraction from a mold, rather than changing the general appearance of a model. Typically the angle supplied is very small.

Replaces surfaces of supplied faces with surfaces tapered by the taper angle about the corresponding edge and direction vector. Note that the corresponding edge must lie on the face to be tapered.

If a face is to be tapered about several edges, these edges must all be supplied, which means that some faces may have to be supplied more than once. This will lead to a topology change

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It can not be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

See the documentation for api_taper_faces for the use of the option “lop_validate”.

Topology Changes:

In addition to the following, refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

The user must supply all edges of the face required to be tapered about, if necessary supplying the same face twice or more. In such cases, new edges will be introduced, splitting the duplicated face into many, each of which is individually tapered about its edge.

Vent faces are added between mergeable faces, when one of them is not being tapered. Vent faces can only be added if there is a face on the original model that shares a vertex with the mergeable edge, does not have the mergeable edge in its boundary and that will be adjacent to the vent faces after the taper. Vent faces can also be added at tangent edges when just one of the two faces that share the edge is being tapered. Note however, that a bent face will only be added if there is no intersection between the surface of the face that is being tapered and the surface if the face that isn’t.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be tapered.

edge–list identifies edges of the body about which the faces from face–list are tapered.

direction specifies the direction vector from which the angle is measured.

angle specifies the draft angle of the taper operation in degrees.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: In addition to the following, refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Only planes, cones, ruled surfaces and previously edge tapered surfaces (provided the same edge and draft direction is used) may be edge tapered.

Example:

```
; lop:edge–taper–faces
; Create a rotated block
(define block1
    (solid:block (position –5 –5 –5)
    (position 5 5 5)))
;; block1
(define transform
    (entity:transform block1 (transform:rotation
    (position 0 0 0) (gvector 1 0 0) 10)))
;; transform
; OUTPUT Original

(define taper (lop:edge–taper–faces (pick:face
    (ray (position 0 0 0) (gvector 1 0 0)))
    (pick:edge (ray (position 5 0 0)
    (gvector 0 0 –1))) (gvector 0 0 1) 20))
;; taper
; OUTPUT result
```

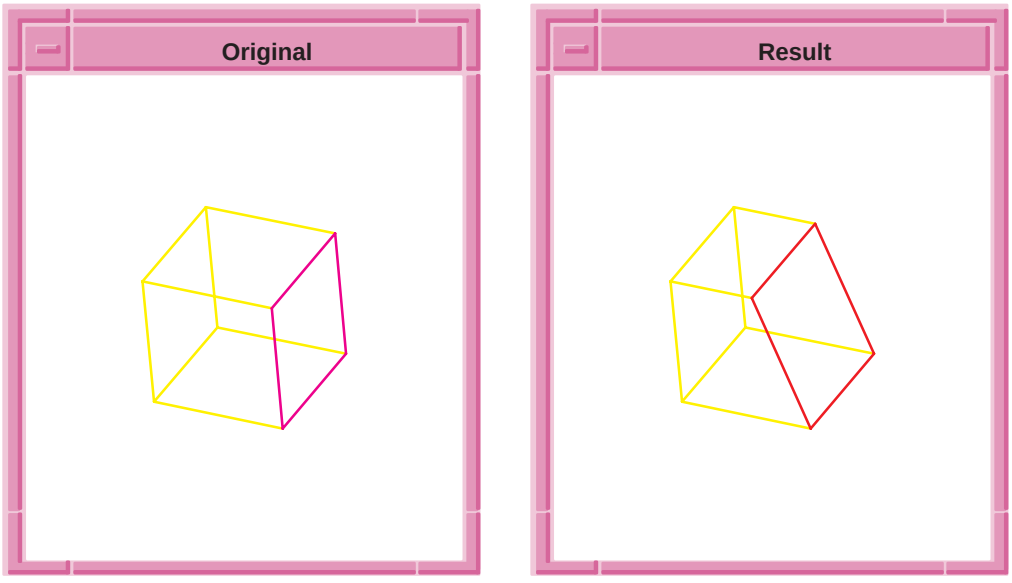


Figure 2-1. lop:edge–taper–faces

```
; Another example of edge taper.
; Clear everything and start clean.
(part:clear)
;; #t
; Create a block
(define block2 (solid:block (position 15 15 0)
    (position 25 25 20)))
;; block2
; OUTPUT Original

; Get a list of the faces
(define face–list (entity:faces block2))
;; face–list
(define face5 (list–ref face–list 5))
;; face5
(define edge–list (entity:edges block2))
;; edge–list
(define edge4 (list–ref edge–list 4))
;; edge4
(define taper (lop:edge–taper–faces face5 edge4
    (gvector 0 0 1) 20))
;; taper
; OUTPUT Result
```

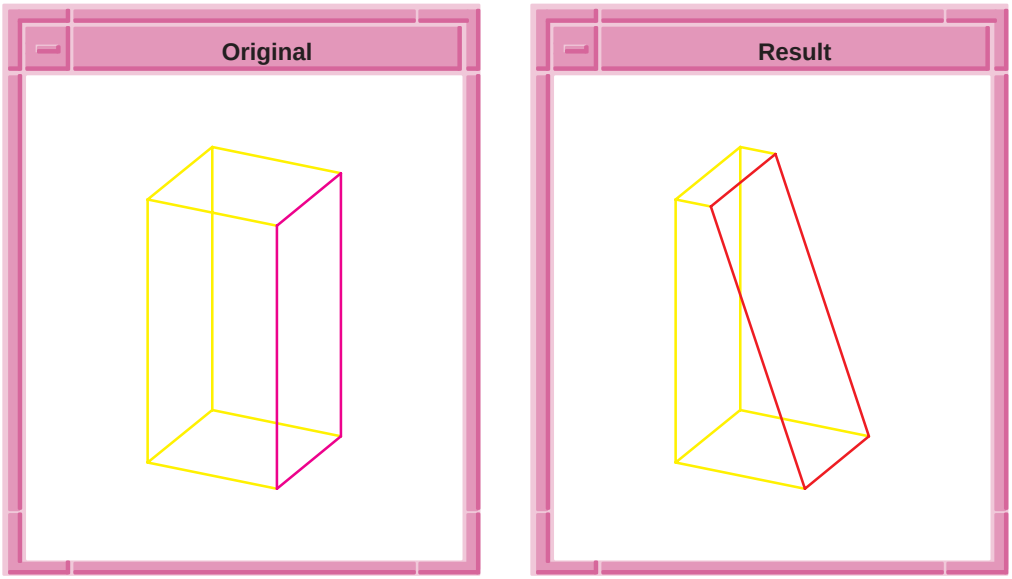
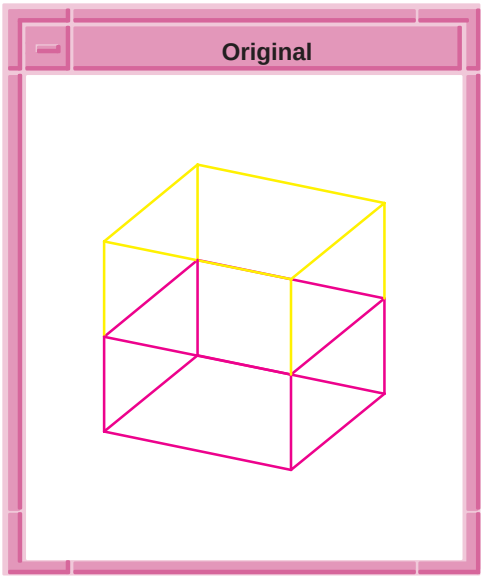


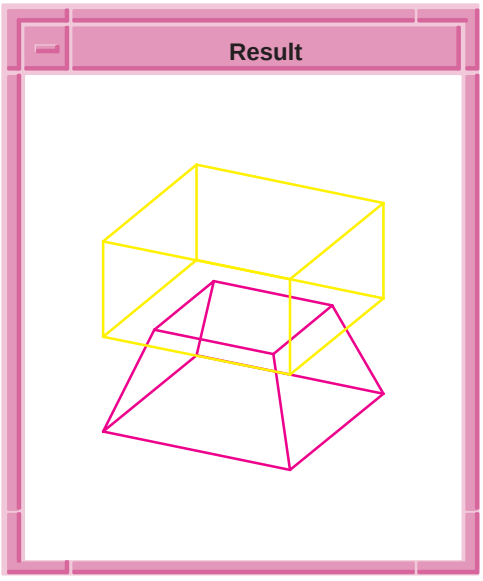
Figure 2-2. lop:edge–taper–faces

```
; A third example of edge taper merging vent faces.
; Define some entities
(part:clear)
;; #t
(define block1 (solid:block
    (position –25 –25 –25) (position 25 25 25)))
;; block1
(define block2 (solid:block
    (position –50 –50 0) (position 50 50 100)))
;; block2
(solid:imprint block1 block2)
;; #t
(entity:delete block2)
;; ()
(option:set ”annotation” #t)
;; #f
(option:set ”unhook_annotations” #f)
;; #t
(define e1 (pick:edge
    (ray (position 0 0 –25) (gvector 1 0 0))))
;; e1
(define f1 (pick:face
    (ray (position 0 0 –20) (gvector 1 0 0))))
;; f1
(define e2 (pick:edge (ray
    (position 0 0 –25) (gvector –1 0 0))))
;; e2
(define f2 (pick:face (ray
    (position 0 0 –20) (gvector –1 0 0))))
;; f2
(define e3 (pick:edge
    (ray (position 0 0 –25) (gvector 0 1 0))))
;; e3
(define f3 (pick:face
    (ray (position 0 0 –20) (gvector 0 1 0))))
;; f3
(define e4 (pick:edge
    (ray (position 0 0 –25) (gvector 0 –1 0))))
;; e4
(define f4 (pick:face
    (ray (position 0 0 –20) (gvector 0 –1 0))))
;; f4
; OUTPUT Original
```

```
(define taper
    (lop:edge-taper-faces (list f1 f2 f3 f4)
    (list e1 e2 e3 e4) (gvector 0 0 1) 20))
;; taper
; OUTPUT Result

; Render to see the final result.
(render)
;; ()
; OUTPUT Rendered
```





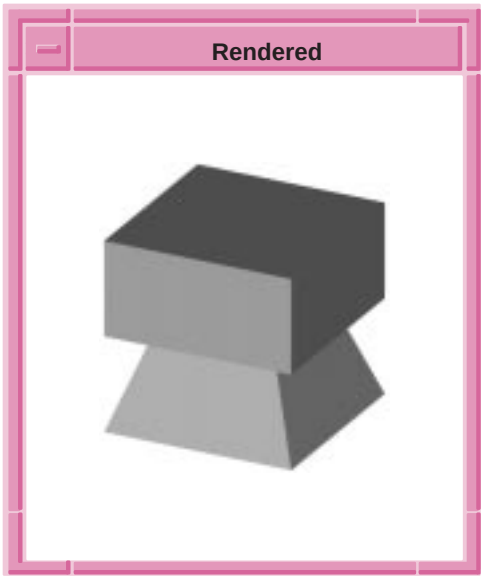


Figure 2-3. lop:edge–taper–faces

lop:move–faces

Scheme Extension: Local Operations

Action: Moves an array of faces through a transform.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_move_faces

Syntax:
```
(lop:move–faces face–list vector–or–transform
    [box–h box–l] [acis–opts])
```

Arg Types:		
	face–list	face | (face ...)
	vector–or–transform	gvector | transform
	box–h	position
	box–l	position
	acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the errors listed for the Scheme extension, lop:tweak–faces.

Transformation must be a rigid motion (i.e., the product of rotations and translations) that is not the identity or the translation vector must be non–zero, or error;

LOP_MOVE_BAD_TRANSFORM

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: Replaces surfaces of supplied faces with surfaces moved by a transform (a translation, when a gvector is supplied.)

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Mergeable edges will be removed unless they have a NO_MERGE_ATTRIB.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be moved.

vector–or–transform specifies the position and direction of the move operation.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Example:

```
; lop:move–faces
; Create a solid block.
(define block1
    (solid:block (position –20 –20 –20)
    (position 20 20 20)))
;; block1
; OUTPUT Original

; List the faces of the block
(entity:faces block1)
;; (#[entity 3 1] #[entity 4 1] #[entity 5 1]
;; #[entity 6 1] #[entity 7 1] #[entity 8 1])
; Move a face on the body
(define move (lop:move–faces (entity 4)
    (gvector 10 10 20)))
;; move
; OUTPUT Result
```

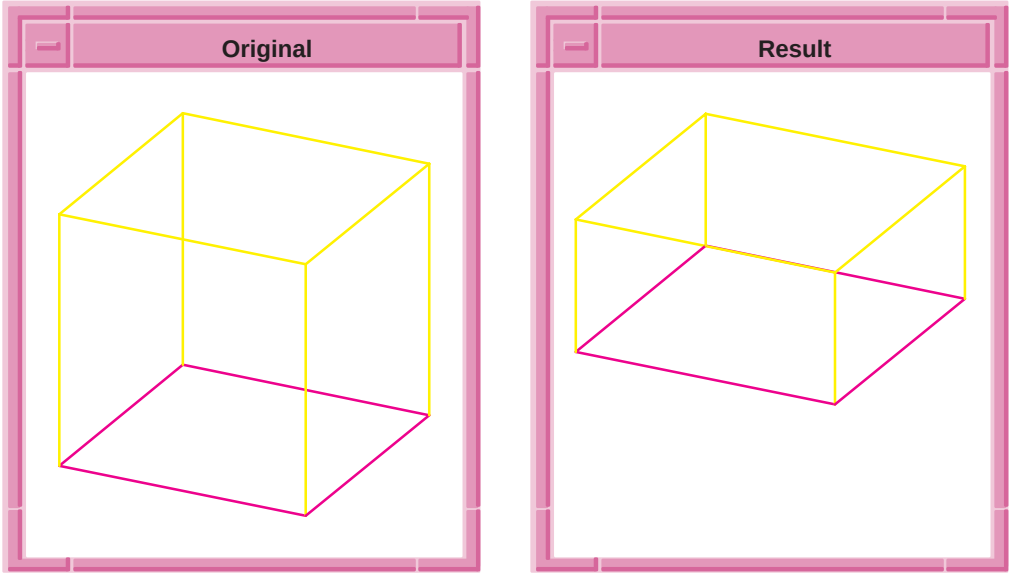


Figure 2-4. lop:move–faces

lop:offset–body

Scheme Extension: Local Operations

Action: Offsets faces of a body.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_offset_body

Syntax: `(lop:offset-body body offset [box-h box-l] [acis-opts])`

Arg Types:		
	body	body
	offset	real
	box–h	position
	box–l	position
	acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the Errors listed for the Scheme extension, lop:tweak–faces.

Valid offset (greater than minus half the body box max side), and not a zero offset (magnitude greater than twice SPAresabs) or error;
LOP_OFF_BAD_OFFSET
Valid body transformation (no shear component) or error;
LOP_BAD_BODY_TRANSFORM

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: Offsets the faces of the supplied body by an offset distance. Faces with radial surfaces which cannot be so offset are removed and the resulting wound healed by the surrounding face surfaces.

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

In addition to the following, refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Mergeable edges will be removed unless they have a NO_MERGE_ATTRIB.

Faces with radial surfaces which cannot be offset by the distance without degenerating or inverting (spheres, cones, blended edges, blended vertices,and tori) are removed and the resulting wound healed by the surrounding face surfaces,before the offset.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

body identifies the body to be offset.

offset specifies the offset distance for all faces.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Example:

```
; lop:offset–body
; Create a solid block.
(define block1
    (solid:block (position –10 –10 –20)
    (position 20 15 20)))
;; block1
; OUTPUT Original

; Offset the body
(define offset (lop:offset–body block1 15))
;; offset
; OUTPUT Result
```

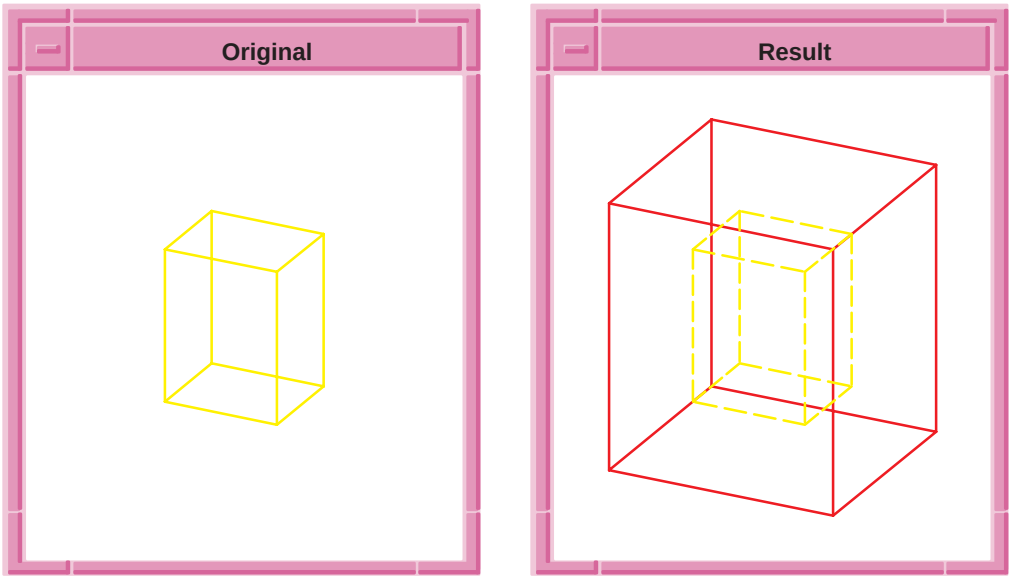


Figure 2-5. lop:offset–body

```
; Example 2
; Clear the previous part
(part:clear)
;; #t
(define block2
    (solid:block (position –25 –25 –25)
    (position 25 25 25)))
;; block2
(define edge1 (blend:const–rad–on–edge
    (pick:edge (ray (position 0 0 0)
    (gvector 1 –1 0))) 10))
;; edge1
(define edge2 (blend:const–rad–on–edge
    (pick:edge (ray (position 0 0 0)
    (gvector 1 0 1))) 15))
;; edge2
(define edge3 (blend:const–rad–on–edge
    (pick:edge (ray (position 0 0 0)
    (gvector 0 –1 1))) 20))
;; edge3
(define vertex1 (blend:on–vertex
    (pick:vertex (ray (position 0 0 0)
    (gvector 1 –1 1))) 20))
;; vertex1
(define blend (blend:network
    (list edge1 edge2 edge3 vertex1)))
;; blend
; OUTPUT Original

(define offset (lop:offset–body block2 –10))
;; offset
; OUTPUT Result
```

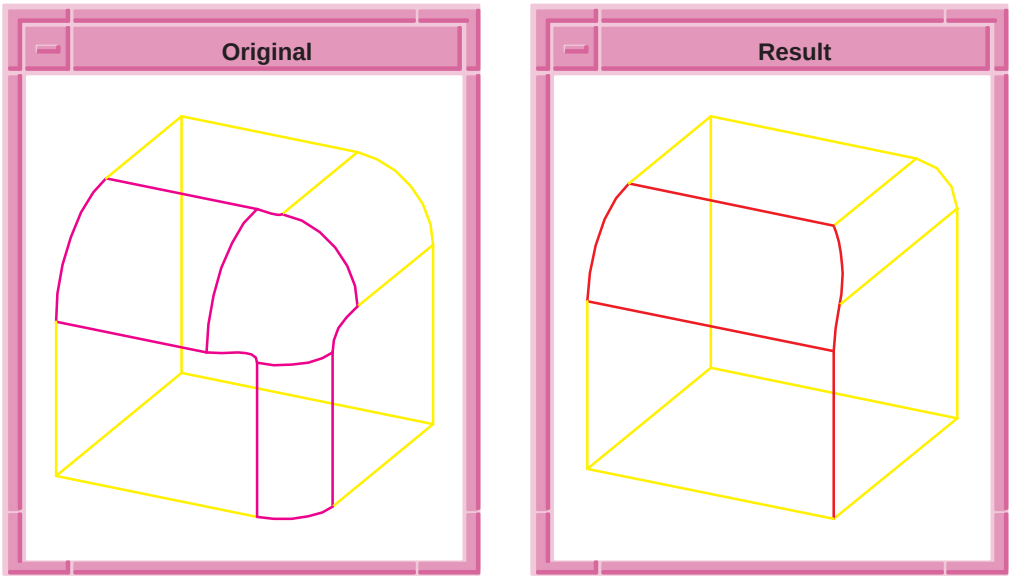


Figure 2-6. lop:offset–body – Example 2

lop:offset–faces

Scheme Extension: Local Operations

Action: Offsets specified faces by a given distance.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_offset_faces

Syntax:
```
(lop:offset–faces face–list offset [box–h box–l]
    [acis–opts])
```

Arg Types:		
	face–list	face | (face ...)
	offset	real
	box–h	position
	box–l	position
	acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the Errors listed for the Scheme extension, lop:tweak–faces.

Valid offset (greater than minus the body box max side), and not a zero offset (magnitude greater than twice SPAresabs) or error;
LOP_OFF_BAD_OFFSET

Valid body transformation (no shear component) or error;
LOP_BAD_BODY_TRANSFORM

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: Replaces surfaces of supplied faces with surfaces offset by offset distance. Radial faces with surfaces which cannot be so offset are removed and the resulting wound healed by the surrounding face surfaces.

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

In addition to the following, refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Mergeable edges will be removed unless they have a NO_MERGE_ATTRIB.

Faces with radial surfaces which cannot be offset by the distance without degenerating or inverting (spheres, cones, blended edges, blended vertices, and tori) are removed and the resulting wound healed by the surrounding face surfaces,before the offset.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be offset.

offset specifies the value of the offset distance.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Example:

```
; lop:offset–faces
; Create a solid block.
(define block1
    (solid:block (position –10 –10 –20)
    (position 5 30 20)))
;; block1
; OUTPUT Original

; Offset a face on the body
(lop:offset–faces (pick:face (ray (position 0 0 0)
    (gvector 0 0 1))) 10)
;; #[entity 2 1]
; OUTPUT Result
```

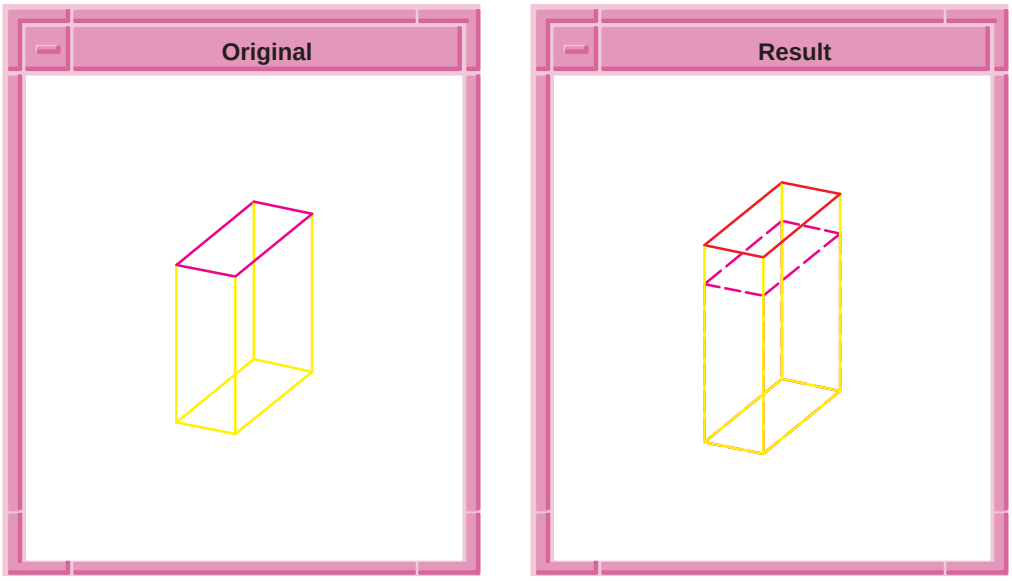


Figure 2-7. lop:offset–faces

lop:offset–specific

Scheme Extension: Local Operations

Action: Offsets specified faces by distances specific to each.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_offset_faces_specific

Syntax:
```
(lop:offset–specific def–face–list def–offset
    spec–face–list spec–offset–list [box–h box–l]
    [acis–opts])
```

Arg Types:

def–face–list	face | (face ...)
def–offset	real
spec–face–list	face | (face ...)
spec–offset–list	real | (real ...)
box–h	position
box–l	position
acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the errors listed for the Scheme extension, lop:tweak–faces.

Valid offset (greater than minus the body box max side), and not a zero offset (magnitude greater than twice SPAresabs) or error;– LOP_OFF_BAD_OFFSET

Valid body transformation (no shear component) or error; LOP_BAD_BODY_TRANSFORM

Note, entities returned in the outcome standard_error_info object are highlighted.

Description: Replaces surfaces of the supplied faces with surfaces offset by an offset distance specific for each face.

Radial faces with surfaces which cannot be so offset are removed and the resulting wound healed by the surrounding face surfaces.

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

In addition to the following, refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Faces with radial surfaces which cannot be offset by the distance without degenerating or inverting (spheres, cones, blended edges, blended vertices, and tori) are removed and the resulting wound healed by the surrounding face surfaces, before the offset.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

def–face–list identifies faces of a body to be offset.

def–offset specifies the value of the offset distance.

spec–face–list identifies faces of a body to be offset. based on the specific face list.

spec–offset–list specifies the value of the offset distance of a specific list of entities.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Example:

```
; lop:offset-specific
; Set the position of the view.
(view:set (position 75 25 50) (position 0 0 0)
    (gvector 0 0 1))
;; #[view 1076820424]
; Set the perspective of the view.
(view:set-perspective #f)
;; #f
(define b1 (solid:block (position -25 -25 -25)
    (position 25 25 25)))
;; b1
; OUTPUT Original
```

```
; Offset bottom and back 10 (default),
; top 20 and front –10 (specific)
(lop:offset–specific
    (list
        (pick:face (ray (position 0 0 0)
        (gvector 0 0 –1)))
        (pick:face (ray (position 0 0 0)
        (gvector –1 0 0)))) 10 (list
        (pick:face (ray (position 0 0 0)
        (gvector 0 0 1)))
        (pick:face (ray (position 0 0 0)
        (gvector 1 0 0))))
    (list 20.0 –10.0))
;; #[entity 2 1]
; OUTPUT Result
```

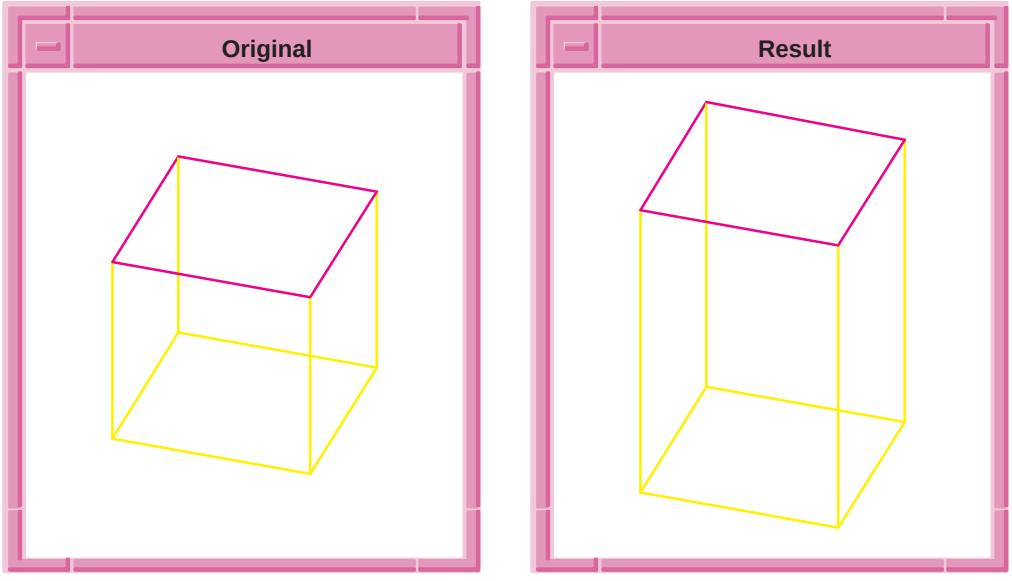


Figure 2-8. lop:offset–specific

lop:remove–attributes

Scheme Extension: Local Operations

Action: Removes all attributes that a local operation attached to EDGEs and VERTEXs contained in an ENTITY.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_remove_lop_attribs

Syntax: (**lop:remove–attributes** entity [acis–opts])

Arg Types:

entity	entity
acis–opts	acis–options

Returns: body

Errors: None

Description: Removes all ATTRIB_LOP_EDGE and ATTRIB_LOP_VERTEX attributes attached to EDGEs and VERTEXs on an ENTITY. These attributes are attached when using the APIs api_tweak_faces_init, api_tweak_fix_edge, and api_tweak_fix_vertex.

Arguments:

entity identifies the entities on a body which will have their attributes removed.

acis–opts specifies options such as versioning and journaling.

Limitations: None

Example:

```
; lop:remove–attributes
; Make a body
; Define options in data structure for sweep:law.
(define opts (sweep:options ”cut_end_off” #t))
;; opts
; Create an edge from the specified law.
(define path (edge:law ”vec(cos(x),
    sin(x),x/5)” 0 20))
;; path
; Create an edge that is part of an ellipse.
(define profile (edge:ellipse (position 1 0 0)
    (gvector 0 1 0) (gvector .2 0 0)))
;; profile
; Create surface by sweeping.
(define sweep (sweep:law profile path opts))
;; sweep
; Delete ”path”.
(entity:delete path)
;; ()
; Select a face.
```

```
(define f (pick:face (ray
    (position 1 .01 0) (gvector 0 –1 0))))
;; f
; Attach lop attributes to edges in the model
(define tweak (lop:tweak–faces–init f f #f))
;; tweak
; Get data structure info. for ”sweep 3”.
(entity:debug sweep 3)
;       1 body record,          32 bytes
;       6 attribute records,    512 bytes
;       1 lump record,          32 bytes
;       1 shell record,         40 bytes
;       3 face records,         132 bytes
;       3 loop records,         96 bytes
;       3 surface records       472 bytes
;       6 coedge records,       264 bytes
;       3 edge records,         216 bytes
;       4 pcurve records,       352 bytes
;       2 vertex records,       48 bytes
;       3 curve records,        336 bytes
;       2 point records,        96 bytes
; Total storage 2640 bytes
;; ”solid body”
; Remove the attribute
(define remove (lop:remove–attributes sweep))
;; remove
; Get adjusted data structure info. for ”sweep 3”.
(entity:debug sweep 3)
;       1 body record,          32 bytes
;       5 attribute records,    216 bytes
;       1 lump record,          32 bytes
;       1 shell record,         40 bytes
;       3 face records,         132 bytes
;       3 loop records,         96 bytes
;       3 surface records       472 bytes
;       6 coedge records,       264 bytes
;       3 edge records,         216 bytes
;       4 pcurve records,       352 bytes
;       2 vertex records,       48 bytes
;       3 curve records,        336 bytes
;       2 point records,        96 bytes
; Total storage 2328 bytes
;; ”solid body”
```

lop:rotate–faces

Scheme Extension: Local Operations

Action: Rotates specified faces through a transform.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_move_faces

Syntax:
```
(lop:rotate–faces face–list point axis angle
    [box–h box–l] [acis–opts])
```

Arg Types:

face–list	face | (face ...)
point	position
axis	gvector
angle	real
box–h	position
box–l	position
acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the Errors listed for the Scheme extension, lop:tweak–faces.

Rotation must be non–zero, or error;
LOP_MOVE_BAD_TRANSFORM

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: Replaces surfaces of supplied faces with surfaces rotated by a transform as supplied by an angle about a point and axis.

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be moved.

point specifies the position for the point of rotation.

axis specifies the axis of rotation.

angle specifies the rotation angle in degrees.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Example:

```
; lop:rotate–faces
; Create a solid block.
(define block1
    (solid:block (position –20 –20 –20)
    (position 20 20 20)))
;; block1
; OUTPUT Original

; Rotate a face on the body
(define rotate (lop:rotate–faces
    (pick:face (ray (position 0 0 0)
    (gvector 0 0 1))) (position 0 0 0)
    (gvector 10 20 10) 10))
;; rotate
; OUTPUT Result
```

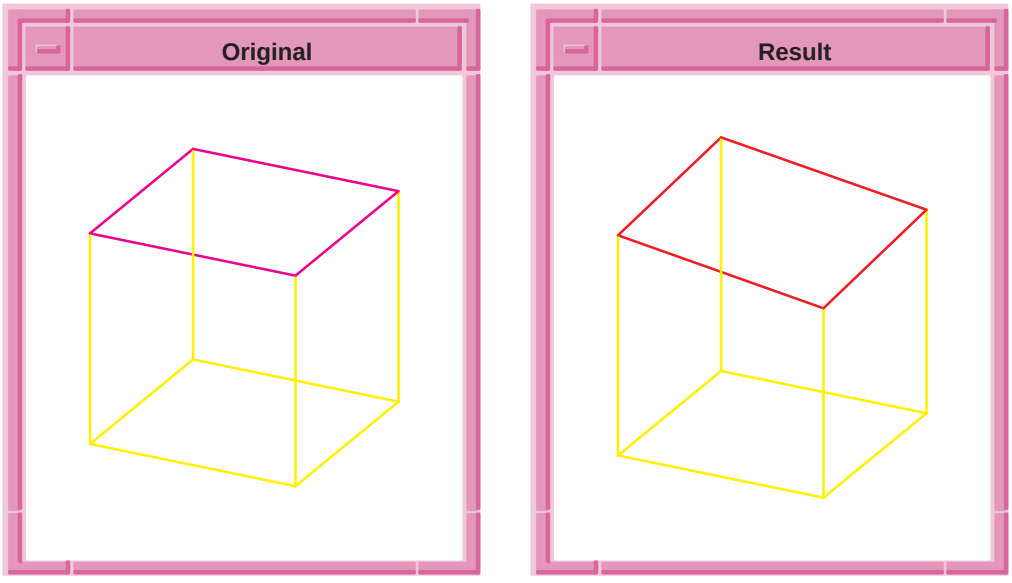


Figure 2-9. lop:rotate–faces

lop:shadow–taper–faces

Scheme Extension: Local Operations

Action: Insert a ruled face which tangentially meets the supplied face, and which covers the region that is in the shadow cast from a light source.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_shadow_taper_faces

Syntax:
```
(lop:shadow–taper–faces face–list direction
    angle [box–h box–l] [acis–opts])
```

Arg Types:		
	face–list	face | (face ...)
	direction	gvector
	angle	real
	box–h	position
	box–l	position
	acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the Errors listed for the Scheme extension, lop:tweak–faces.

Some of the following errors result in an ENTITY, which indicates where the error occurs, being highlighted. The ENTITY type follows the error message below.

Valid direction must be supplied, or error;
LOP_TAP_BAD_NORMAL
Valid angle (at least 0 and less than 90 degrees) must be supplied or error;
LOP_TAP_BAD_ANGLE
Surface must be able to be tapered as requested, or error;
LOP_TAP_NO_SURF

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: The purpose of a shadow taper is to facilitate extraction from a mold rather than changing the general appearance of a model. Typically the angle supplied is very small.

This extension inserts a ruled face which tangentially meets the supplied face, and which covers the region in the shadow that is cast from a light source. The light source is directed from an infinite point and is at a given angle from the supplied draft direction.

This extension replaces regions of the face to be shadow tapered where the angle between the normal and the draft vector is greater than the complement of the given angle (i.e., greater than 90 – given angle) by ruled faces where the angle between the draft vector and the normal is exactly the complement of the given angle. Thus, the ruled faces meet the face to be tapered tangentially and are bounded by the intersections with either neighboring faces or the same face. The regions to be replaced are referred to as being “in shadow” with respect to the draft vector and the given angle.

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It can not be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

In addition to the following, refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Almost every shadow taper will result in a topology change as the parts of the body that are ”in shadow” are replaced by ruled faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be tapered.

direction specifies the direction vector pointing towards the light source.

angle specifies the draft angle of the taper operation in degrees.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: In addition to the following, refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

No more than one face that is in the shadow can be deleted.

In general, there must be an intersection between the new face and either a face that is adjacent to the face to be tapered or a face that is “adjacent–but–one” to the face to be tapered.

Example:

```
; lop:shadow–taper–faces
; Create a solid block
(define block1
    (solid:block (position –25 –25 –10)
    (position 25 25 10)))
;; block1
; Create a solid cone
(define cone1
    (solid:cone (position –20 0 10)
    (position 20 0 10) 20 10))
;; cone1
; Unite the cone and block
(define unite (bool:unite block1 cone1))
;; unite
; OUTPUT Original
```

```
; Shadow taper a face on the body
(define taper (lop:shadow–taper–faces
    (pick:face (ray (position 0 0 0)
    (gvector 0 0 1))) (gvector 0 0 1) 20))
;; taper
; OUTPUT Result
```

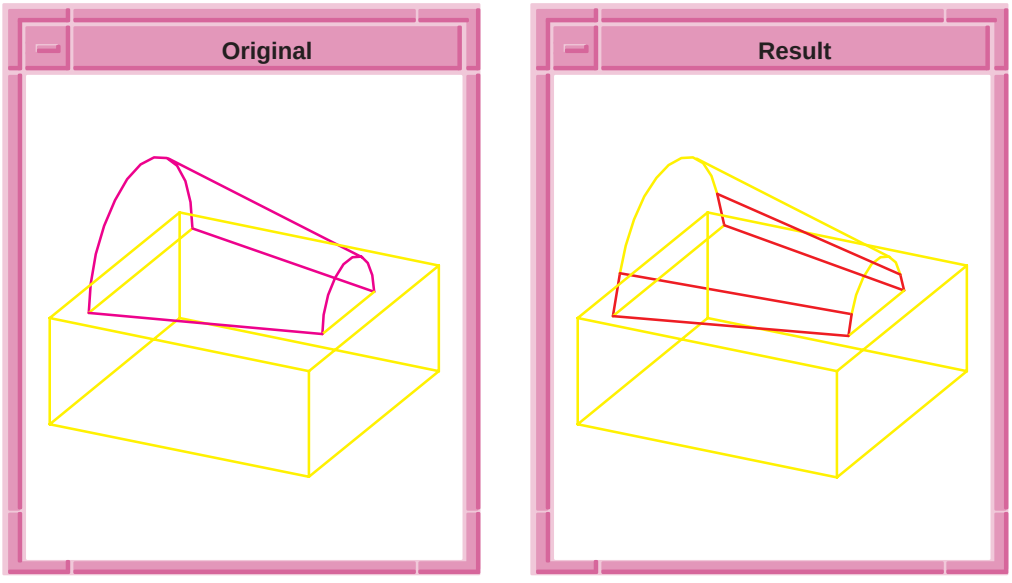


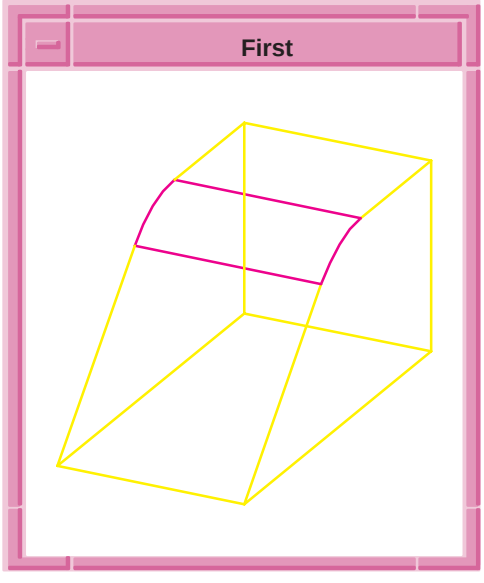
Figure 2-10. lop:shadow–taper–faces

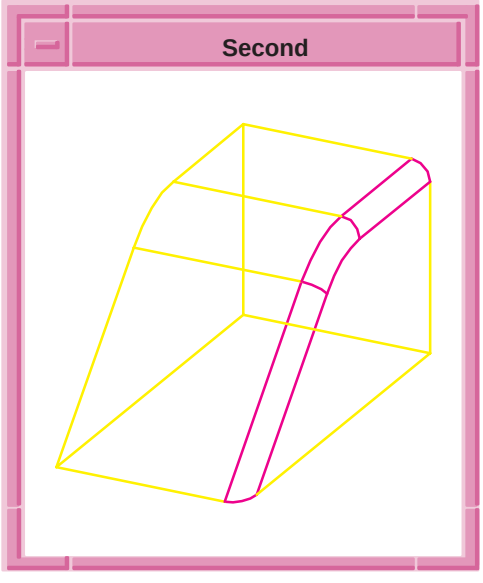
```
; Another example of lop:shadow–taper–faces
; this example illustrates this command on multiple
;   complex blends.
(part:clear)
;; #t
(define b1 (solid:block
    (position –25 –50 –25) (position 25 50 25)))
;; b1
(define b2 (solid:block
    (position –100 –100 0) (position 100 100 200)))
;; b2
(define transform (entity:transform b2
    (transform:rotation
    (position 0 0 0) (gvector 1 0 0) 45)))
;; transform
(define transformb2 (entity:transform b2
    (transform:translation (gvector 0 0 25))))
;; transformb2
(define subtract (bool:subtract b1 b2))
;; subtract
(define edge1 (blend:const–rad–on–edge
    (pick:edge (ray (position 0 0 0)
    (gvector 0 0 1))) 30))
;; edge1
(define network
    (blend:network (blend:get–network edge1)))
;; network
; OUTPUT First

(define edge2 (car (blend:const–rad–on–edge
    (list (pick:edge (ray (position 25 20 –20)
    (gvector 0 0 1))) (pick:edge (ray
    (position 25 0 –20) (gvector 0 0 1)))
    (pick:edge (ray (position 25 –20 –20)
    (gvector 0 0 1)))) 5)))
;; edge2
(define network2 (blend:network
    (blend:get–network edge2)))
;; network2
(option:set ”annotation” #t)
;; #f
; OUTPUT Second
```

```
(option:set "unhook_annotations" #f)
;; #t
(lop:shadow-taper-faces (list (pick:face
    (ray (position 24 -20 -20) (gvector 0 0 1)))
    (pick:face (ray (position 24 0 -20)
    (gvector 0 0 1))) (pick:face (ray
    (position 24 20 -20) (gvector 0 0 1))))
    (gvector 0 0 1) 30)
;; #[entity 5 1]
; OUTPUT Third

; Render the image
(render)
;; ()
; OUTPUT Rendered
```





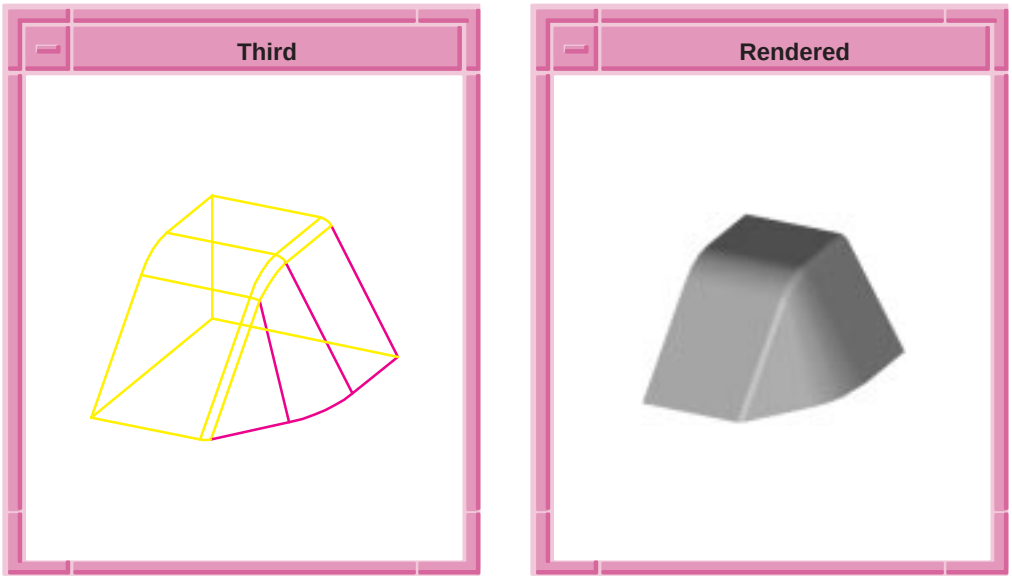


Figure 2-11. lop:shadow–taper–faces

lop:sweep–more

Scheme Extension: Local Operations, Sweeping

Action: Sweeps an array of faces along a path defined by the faces adjacent to the given faces.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_sweep_more

Syntax:
```
(lop:sweep–more face–list distance [box–l box–h]
    [acis–opts])
```

Arg Types:		
	face–list	face | (face ...)
	distance	real
	box–l	position
	box–h	position
	acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the Errors listed for the Scheme extension, lop:tweak–faces.

Some of the following errors result in an entity, which indicates where the error occurs, being highlighted. The entity type follows the error message below.

Distance must be non–zero, or error;
LOP_MOVE_BAD_TRANSFORM

Rail and path information on adjacent swept face must be good, or error;
LOP_SWP_BAD_RAIL_OR_PATH (FACE *)

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: Replaces surfaces of supplied faces with surfaces moved along the sweep path. The sweep path is determined by a face that is not in that set of faces given but is adjacent to at least one of those faces. Note, this face must determine the sweep path, i.e., it must be a cone, cylinder, plane, torus, sweep_spl_sur, rot_spl_sur, sum_spl_sur, or an offset of one of these surfaces.

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be moved.

distance is the distance along the sweep path that the faces are moved.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: None

Example:

```
; lop:sweep–more
; Make a spring
(define path
    (edge:law "vec(cos(x),sin(x),x/10)" 0 10))
;; path
; Define a profile
(define profile
    (edge:ellipse (position 1 0 0)
    (gvector 0 1 0) (gvector .2 0 0)))
;; profile
(define sweep (sweep:law profile path))
;; sweep
; sweep => #[entity 4 1]
; Zoom to see the model
(zoom–all)
;; #[view 1076896136]
(render:rebuild)
;; ()
; OUTPUT Original and OUTPUT Rendered Original

; Pick the face to sweep more
(define f (pick:face (ray (position 1 0 0)
    (gvector 0 –1 0))))
;; f
(define sweep (lop:sweep–more f 1))
;; sweep
(render)
;; ()
; OUTPUT Result and OUTPUT Rendered Result
```

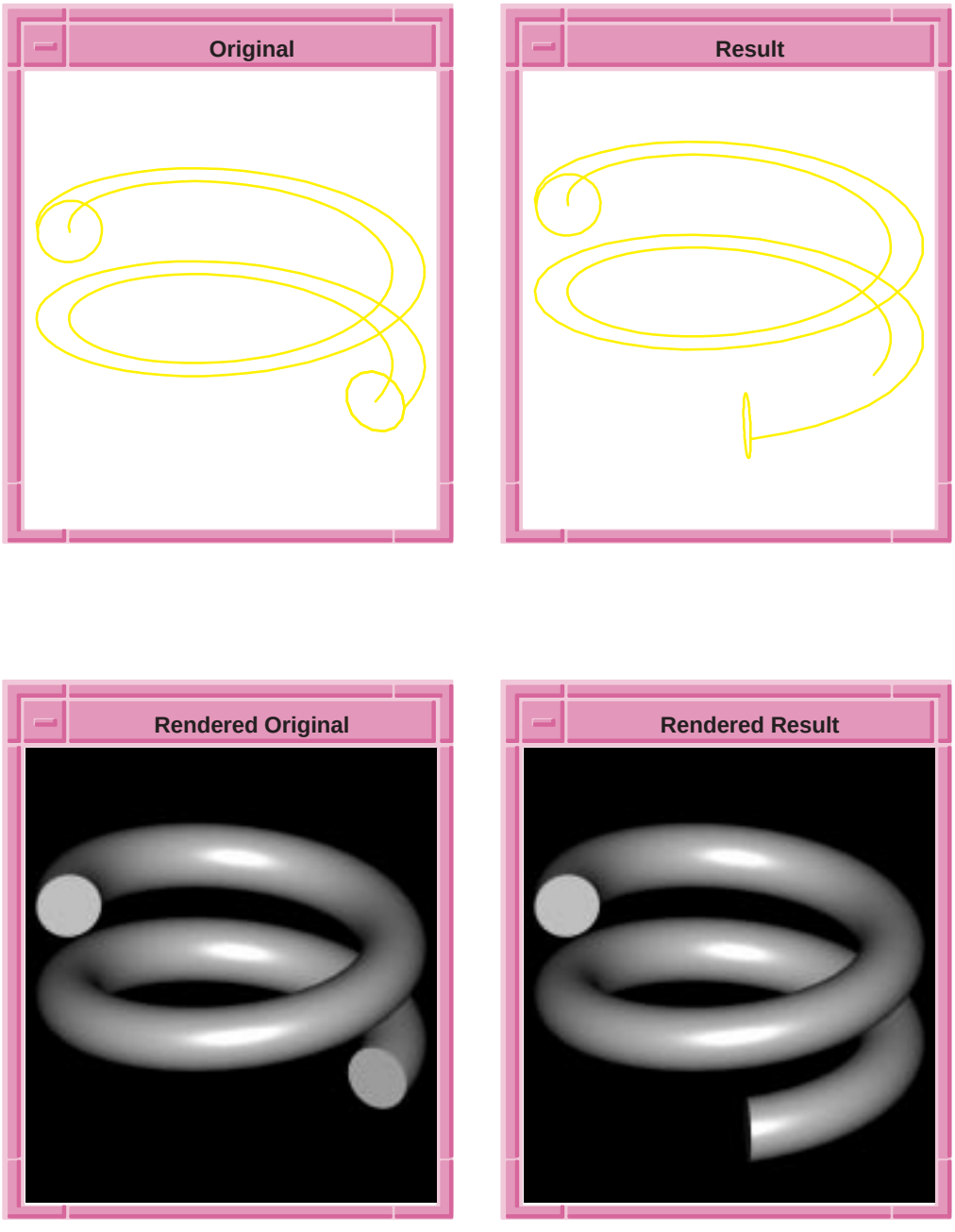


Figure 2-12. lop:sweep–more

lop:taper–faces

Scheme Extension: Local Operations

Action: Tapers an array of faces about a point and a supplied draft vector by a given draft angle.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_taper_faces

Syntax:
```
(lop:taper–faces face–list point normal angle
    [box–h box–l] [acis–opts])
```

Arg Types:

face–list	face | (face ...)
point	position
normal	gvector
angle	real
box–h	position
box–l	position
acis–opts	acis–options

Returns: body

Errors: In addition to the following, refer to the Errors listed for the Scheme extension, lop:tweak–faces.

Some of the following errors result in an entity, which indicates where the error occurs, being highlighted. The entity type follows the error message below.

Valid direction must be supplied, or error;
LOP_TAP_BAD_NORMAL
Valid angle (at least 0 and less than 90 degrees) must be supplied or error;
LOP_TAP_BAD_ANGLE
Surface must be able to be tapered as requested, or error;
LOP_TAP_NO_SURF
Normal of face to be tapered should not be parallel to draft vector, or error;
LOP_TAP_NO_SURF_FACE
Must be able to find a face adjacent to vent face if a vent face is inserted, or error;
LOP_TAP_NO_ADJ_FACE

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: This extension tapers the face(s) specified by face–list by the supplied draft angle about an axis defined by the intersection between the plane of the face and a ”taper plane”. The taper plane is defined by a point and normal given as arguments to the extension. The direction of the normal defines the direction of the angle, i.e., the resulting angled plane will slope in the direction of the draft plane normal. The taper plane need not intersect the face to be tapered, and even when it does, this intersection need not be an edge of the body (unlike in edge taper).

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

See the documentation for api_taper_faces for the use of the option “lop_validate”.

Topology Changes:

In addition to the following, refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Vent faces are added between mergeable faces, when one of them is not being tapered. Vent faces can only be added if there is a face on the original model that shares a vertex with the mergeable edge, does not have the mergeable edge in its boundary and that will be adjacent to the vent faces after the taper. Vent faces can also be added at tangent edges when just one of the two faces that share the edge is being tapered. However, a vent face will only be added if there is no intersection between the surface of the face that is being tapered and the surface of the face that isn’t.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be moved.

point specifies a position on the taper plane.

normal specifies the normal of the taper plane at the point specified.

angle specifies the rotation angle in degrees.

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Refer to the Limitations listed for the Scheme extension, lop:tweak–faces.

Only planes, cones, rule surfaces, and previously plane-tapered surfaces (provided the same taper plane is used) may be plane tapered.

Example:

```
; lop:taper–faces
; Create a solid block.
(define block1
    (solid:block (position –30 –5 –10)
    (position 20 20 10)))
;; block1
; OUTPUT Original

; Taper a face on the body
(define taper (lop:taper–faces
    (pick:face (ray (position 0 0 0)
    (gvector 0 0 1))) (position 0 0 0)
    (gvector 10 20 10) 10))
;; taper
; OUTPUT Result
```

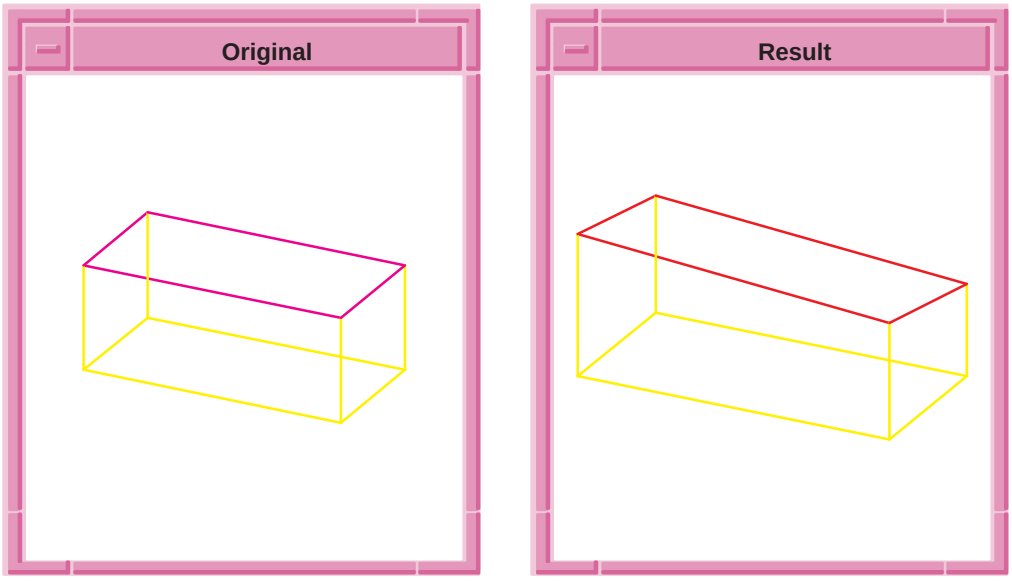


Figure 2-13. lop:taper–faces

```
; Example 2
; Clear the screen to prepare for the next example.
(part:clear)
;; #t
(define block2
    (solid:block (position –25 –25 –10)
    (position 25 25 10)))
;; block2
(define blend (solid:blend–edges
    (pick:edge (ray
    (position 0 0 0) (gvector 1 –1 0))) 20))
;; blend
; OUTPUT Original

(lop:taper–faces (list (pick:face
    (ray (position 0 0 0) (gvector 1 0 0)))
    (pick:face (ray (position 0 0 0)
    (gvector 0 –1 0))) (pick:face
    (ray (position 0 0 0) (gvector 1 –1 0))))
    (position 0 0 –10) (gvector 0 0 1) 45)
;; #[entity 4 1]
; OUTPUT Result
```

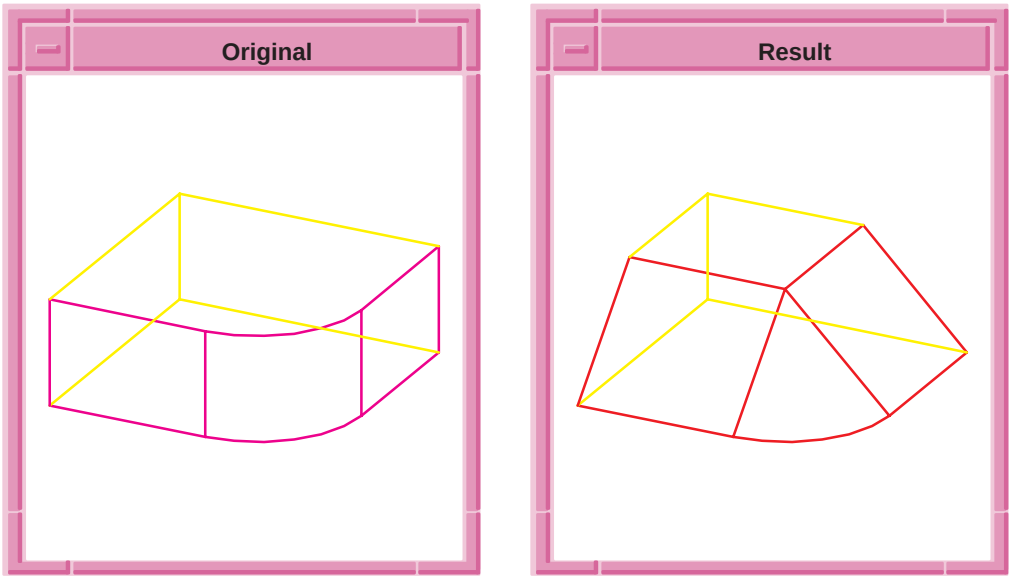


Figure 2-14. lop:taper–faces

lop:tweak–faces

Scheme Extension: Local Operations

Action: Tweaks specified faces.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_tweak_faces

Syntax:
```
(lop:tweak–faces face–list tool–list sense–list
    [box–h box–l] [acis–opts])
```

Arg Types:		
	face–list	face | (face ...)
	tool–list	face | (face ...)
	sense–list	(boolean ...)
	box–h	position
	box–l	position
	acis–opts	acis–options

Returns: body

Errors: Some of the following errors result in an ENTITY, which indicates where the error occurs, being highlighted. The ENTITY type follows the error message below.

At least one face must be supplied, or error;
LOP_TWK_NO_FACE
Faces must be valid, non–duplicate and from the same body, or error;
LOP_TWK_BAD_FACE
Tool surfaces must be valid or error;
LOP_TWK_BAD_SURFACE
Body must be manifold or error;
LOP_TWK_NON_MANIFOLD
Body must be solid or error;
LOP_TWK_FREE_EDGE
New surface geometries must intersect consistently, or errors
LOP_TWK_NO_EDGE no solution for an edge
LOP_TWK_NO_VERT no solution for a vertex
Internal algorithmic problems produce error;
LOP_TWK_INTERNAL
Unpermitted topology changes, if detected, will produce the error;
LOP_TWK_TOPOL_CHANGE
Box, if supplied must be valid, or error;
LOP_BAD_BOX

Note that entities returned in the outcome standard_error_info object are highlighted.

Description: Replaces surfaces of supplied faces with supplied surfaces, taking the supplied reverse flags into consideration. Choose reverse flags carefully, as resulting body is not checked.

The optional intersection box limits the size of intersections between surfaces which might otherwise be very long. It cannot be used to choose faces. Its main purpose is to speed up complicated cases where the intersection curves might be very long, thus improving performance. The box must contain the final faces and if omitted defaults to the size box.

Topology Changes:

The topology changes performed by tweak–faces on the tweaked faces and their neighbors are:

Mergeable edges and vertices on the supplied faces will be merged out.

Edges and vertices between faces that are tweaked to the same geometry will also be merged out.

Two edge vertices on the supplied faces between edges on the same geometrical non analytic intersection curve, which are not presently mergeable elsewhere, will be merged out by lop:tweak–faces if the option lop_merge_vertex is set on, which is the default setting.

Vertices with more than three edges may split into two or more vertices during a local operation. Single inverted faces needed to fill any gap in the body will be made (if two or more inverted faces are needed the operation will fail).

Isolated single edge loops may degenerate exactly to a point or be removed altogether and their faces will be deleted when appropriate.

Edges in multi edge loops may degenerate exactly to a point and will be removed.

Where an edge exactly degenerates to a point in a multi edge loop, the edges either side will be detected becoming fully or partially coincident, and the loop repaired by splitting it into valid and degenerate loops. Degenerate loops will be deleted, and their faces when appropriate.

When faces have been deleted, their shells and lumps will be repartitioned and split if necessary.

The option lop_ff_int switches on face face intersection checking on the result body. The default for the option is off. If the option is switched on and a face face intersection is found in the body, the operation fails.

The option lop_repair_self_int switches on repairing of self intersections on the resulting body. The default for the option is off. If the option is switched on api_repair_body_self_ints is called at the end of successful tweaks and supplied with the changed body faces.

If repairing self intersections is required, the options lop_check_invert and lop_ff_int are disabled internally so that self–intersection checks are avoided and a possibly invalid body is passed to api_repair_body_self_ints with no error raised.

Geometry Changes:

The geometry changes performed by tweak–faces on the tweaked faces and their neighbors are:

pipe_spl_sur spline surfaces will be converted to rb_blend_spl_sur spline surfaces in the supplied faces and their neighbors if the option lop_convert_pipe is set on, which is the default setting.

Arguments:

face–list identifies faces of a body to be moved.

tool–list specifies the new faces created by the tweak.

sense–list flags to reverse new surfaces; #f = leave as–is, #t = reverse

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Body must be manifold and solid.

No other topology changes than those documented above are permitted. Tweak faces does not extend the geometry of the tool surfaces or their neighbors (unlike the other local operations which do).

Example:

```
; lop:tweak–faces
; Create a solid block.
(define block1
    (solid:block (position –20 –20 –20)
    (position 20 20 20)))
;; block1
; Create a tool face
(define face1 (face:plane (position –10 –10 0) 20 20
    (gvector 0 0 1)))
;; face1
; OUTPUT Original

; Tweak face of the body
(define tweak (lop:tweak–faces (pick:face
    (ray (position 10 0 0)
    (gvector 0 0 1))) face1 #f))
;; tweak
; OUTPUT Result
```

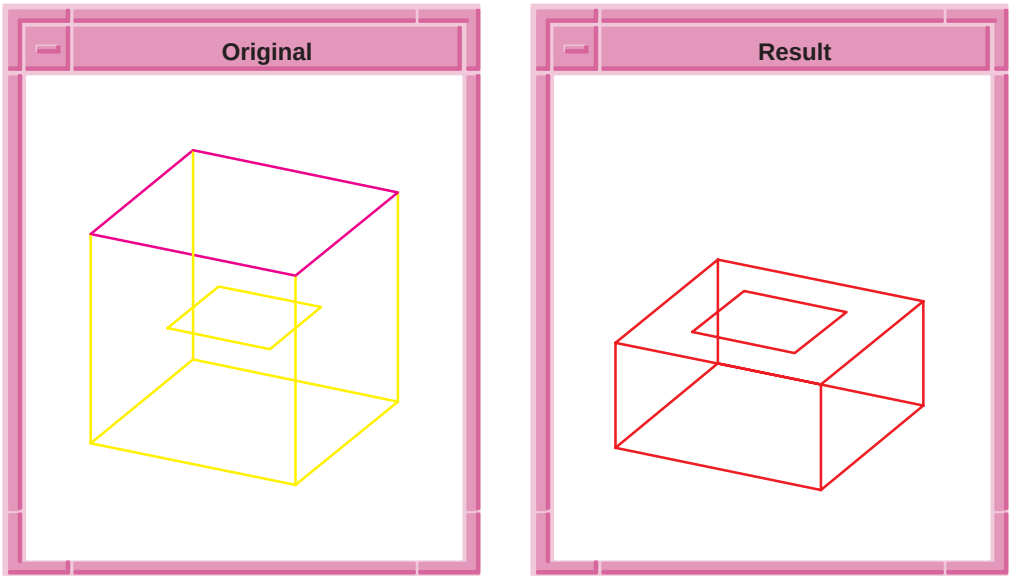


Figure 2-15. lop:tweak–faces

lop:tweak–faces–extend

Scheme Extension: Local Operations

Action: Tweaks specified faces, extending surfaces as necessary.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_tweak_extend_faces, api_tweak_faces

Syntax:
```
(lop:tweak–faces–extend face–list tool–list
    sense–list [box–h box–l] [acis–opts])
```

Arg Types:		
	face–list	face | (face ...)
	tool–list	face | (face ...)
	sense–list	(boolean ...)
	box–h	position
	box–l	position
	acis–opts	acis–options

Returns: body

Errors: Refer to the errors listed for the Scheme extension lop:tweak–faces.

Description: This command extends the surfaces as necessary before doing the tweak. Refer to the description listed for the Scheme extension, lop:tweak–faces, for a full description.

The curve's underlying edges between two faces that are not directly involved in the tweak may also be extended as necessary.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be tapered.

tool–list specifies the new faces created by the tweak.

sense–list flags to reverse new surfaces; #f = leave as–is, #t = reverse

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: Body must be manifold and solid.

Example:

```
; lop:tweak–faces–extend
; Define a wiggle
(define wiggle (solid:wiggle 20 20 20 1 –1 2 –2))
;; wiggle
; Define a face
(define f1 (pick:face (ray (position 0 0 0)
    (gvector 1 0 0))))
;; f1
; Define another face
(define f2 (face:plane
    (position 30 –10 –10) 20 20 (gvector –1 0 0)))
;; f2
; OUTPUT Original
```

```
(define tweak (lop:tweak–faces–extend f1 f2 #t))
;; tweak
; OUTPUT Result
```

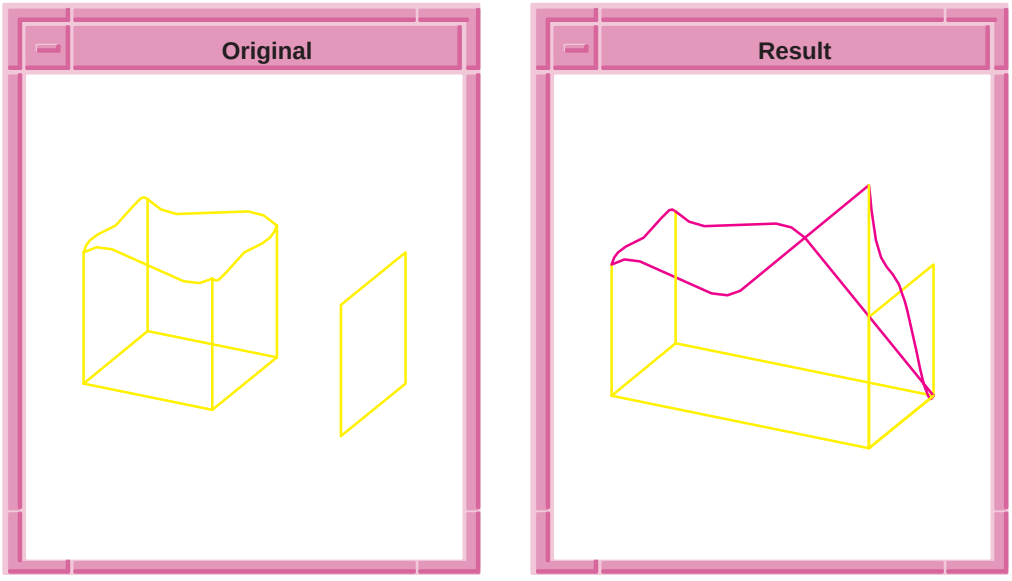


Figure 2-16. lop:tweak–faces–extend

lop:tweak–faces–init

Scheme Extension: Local Operations

Action: Tweaks specified faces attaching attributes to edges involved in the solution. Returns those edges with more than one possible solution.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_tweak_faces_init

Syntax:
```
(lop:tweak–faces–init face–list tool–list sense–list
    [box–h box–l] [acis–opts])
```

Arg Types:		
	face–list	face | (face ...)
	tool–list	face | (face ...)
	sense–list	(boolean ...)
	box–h	position
	box–l	position
	acis–opts	acis–options

Returns: edge | (edge ...)

Errors: Refer to the errors listed for the Scheme extension lop:tweak–faces.

Description: This extension completes the initial stage of a tweak where attributes are added to edges. Returns a list of edges that have more than one possible solution so the user may pick which solution the tweak uses in the next stage. To complete the tweak, use api_tweak_faces, which will continue the tweak without duplicating the work already done.

box–l and box–h arguments are used to limit any geometrical intersections and so improve performance. Limiting the intersection improves performance. The box must contain the final faces and, if omitted, defaults to the size box.

Refer to the function lop:tweak–faces for a complete description.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

face–list identifies faces of a body to be moved.

tool–list specifies the new faces created by the tweak.

sense–list flags to reverse new surfaces; #f = leave as–is, #t = reverse

box–h specifies one position defining a diagonal box for an intersection limit.

box–l specifies one position defining a diagonal box for an intersection limit.

acis–opts specifies options such as versioning and journaling.

Limitations: None

Example:

```
; lop:tweak–faces–init
; Make a spring
(define opts (sweep:options ”cut_end_off” #t))
;; opts
(define path (edge:law ”vec(cos(x), sin(x),x/5)”
    0 20))
;; path
(define profile (edge:ellipse (position 1 0 0)
    (gvector 0 1 0) (gvector .2 0 0)))
;; profile
(define sweep (sweep:law profile path opts))
;; sweep
; Zoom to see the model
(define zoom (zoom–all))
;; zoom
; Pick a tweak face
(define f (pick:face (ray (position 1 .01 0)
    (gvector 0 –1 0))))
;; f
; Tweak initialize the face to its same geometry
(define edge (car (lop:tweak–faces–init f f #f)))
;; edge
; Finish the tweak
(define tweak f f #f)
;; tweak
```

lop:tweak–pick–edge–solution

Scheme Extension: Local Operations

Action: Specifies an edge solution for a subsequent tweak to use.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_tweak_pick_edge_solution

Syntax: (**lop:tweak–pick–edge–solution** edge solution)

Arg Types:		
	edge	edge
	solution	integer | curve | edge

Returns: body

Errors: None

Description: After a call to api_tweak_faces_init, attributes containing edge solution curves are attached to edges involved in the tweak. This function allows the user to specify which of multiple solutions that a subsequent tweak will use. Note that it is possible to choose a solution that forces the tweak to fail or to make a bad model. It is also possible to specify incompatible solutions. Therefore, the user must take care when using this function.

To query the possible solutions on an edge, use api_tweak_query_edge_solutions.

To complete the tweak, use api_tweak_faces, which will continue the tweak using the specified edge solution.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

edge identifies the boundary for the tweak.

solution identifies edge for the tweak.

Limitations: None

Example:

```
; lop:tweak–pick–edge–solution
; Define the sweep options
(define opts (sweep:options ”cut_end_off” #t))
;; opts
(define path(edge:law ”vec(cos(x), sin(x),x/5)”
    0 20))
;; path
(define profile(edge:ellipse (position 1 0 0)
    (gvector 0 1 0) (gvector .2 0 0)))
;; profile
(define sweep(sweep:law profile path opts))
;; sweep
; Zoom to see the model
(zoom–all)
;; #[view 1076896360]
(render:rebuild)
;; ()
; Pick a tweak face
(define f (pick:face (ray (position 1 .01 0)
    (gvector 0 –1 0))))
;; f
; Tweak initialize the face to its same geometry
(define edge (car (lop:tweak–faces–init f f #f)))
;; edge
; Pick the fourth solution
; OUTPUT Original

(define tweak1 (lop:tweak–pick–edge–solution edge 4))
;; tweak1
; Finish the tweak using the picked solution
(define tweak2 (lop:tweak–faces f f #f))
;; tweak2
; OUTPUT Result

(render)
;; ()
; OUTPUT Rendered
```

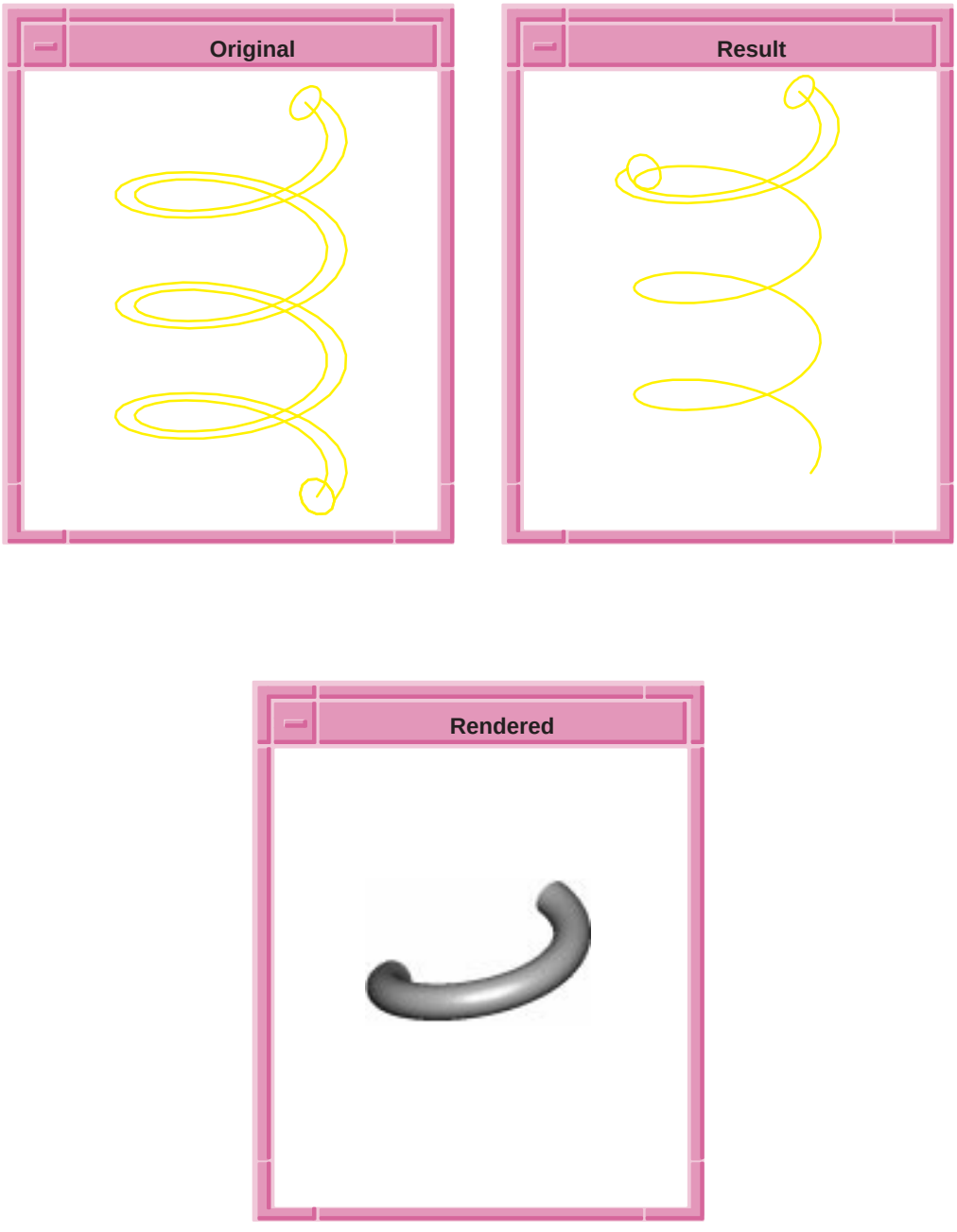


Figure 2-17. lop:tweak–pick–edge–solution

lop:tweak–query–edge–solutions

Scheme Extension: Local Operations

Action: Returns a list of edges built on curves that are possible solutions for an edge's geometry in a tweak.

Filename: lop/lop_scm/lop_scm.cxx

APIs: api_part_note_state, api_part_start_state, api_tweak_query_edge_solutions

Syntax: `(lop:tweak-query-edge-solutions edge)`

Arg Types: edge edge

Returns: edge | (edge ...)

Errors: If the edge does not have a LOP_EDGE_ATTRIB attribute, i.e. the edge was not involved in the initial stages of a tweak LOP_TWK_NO_EDGE.

Description: After a call to api_tweak_faces_init, attributes containing edge solution curves are attached to edges involved in the tweak. This function allows the user to obtain a list of the possible solutions for an edge. Note that the edges returned in the edge list are created on the heap and it is the caller's responsibility to lose them. These edges simply wrap the curves returned by api_trweak_query_edge_solutions for display purposes.

To select a possible solution on the edge, use api_tweak_pick_edge_solution.

To complete the tweak, use api_tweak_faces, which will continue the tweak using the specified edge solution.

Topology Changes:

Refer to the topology changes listed for the Scheme extension, lop:tweak–faces.

Geometry Changes:

Refer to the geometry changes listed for the Scheme extension, lop:tweak–faces.

Arguments:

edge identifies the boundary for the tweak.

Limitations: None

Example:

```
; lop:tweak–query–edge–solutions
; Define sweep options
(define opts (sweep:options ”cut_end_off” #t))
;; opts
(define path(edge:law ”vec(cos(x), sin(x),x/5)”
    0 20))
;; path
(define profile(edge:ellipse (position 1 0 0)
    (gvector 0 1 0) (gvector .2 0 0)))
;; profile
(define sweep(sweep:law profile path opts))
;; sweep
; Zoom to see the model
(zoom–all)
;; #[view 1076896360]
(render:rebuild)
;; ()
; Pick a tweak face
(define f (pick:face (ray (position 1 .01 0)
    (gvector 0 –1 0))))
;; f
; Tweak initialize the face to its same geometry
(define edge (car (lop:tweak–faces–init f f #f)))
;; edge
; OUTPUT Original

; Look at the possible edge solutions
(define solns (lop:tweak–query–edge–solutions edge))
;; solns
; OUTPUT Possible Solutions

; Use the fifth solution to finish the tweak
(define soln (list–ref solns 5))
;; soln
(define tweak–pick
    (lop:tweak–pick–edge–solution edge soln))
;; tweak–pick
(define tweak–faces (lop:tweak–faces f f #f))
;; tweak–faces
; #[entity 4 1]
; OUTPUT Result

; Render the output
(render)
;; ()
; OUTPUT Rendered
```

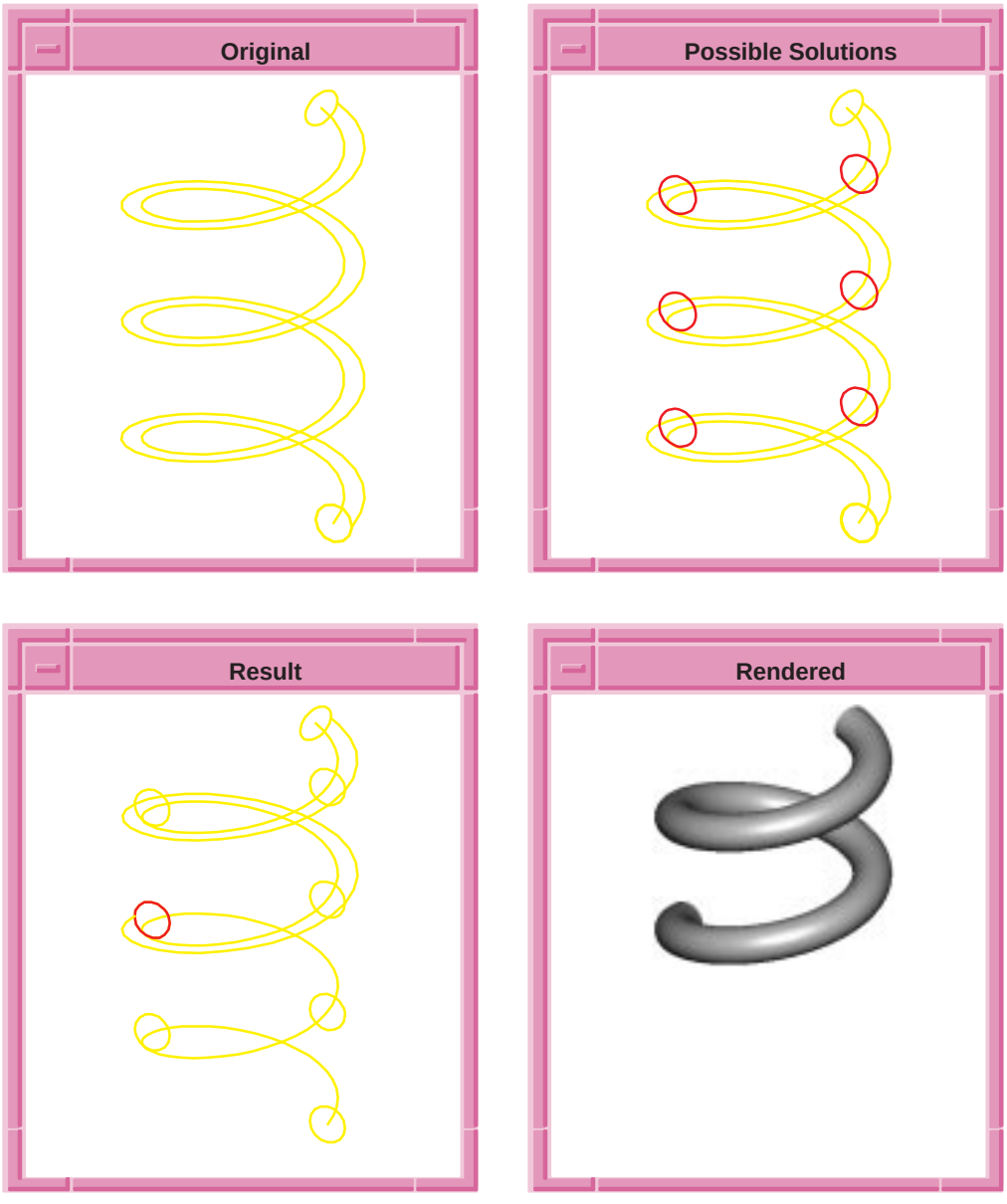


Figure 2-18. lop:tweak–query–edge–solutions