Chapter 3.
Functions

Topic: Ignore

The function interface is a set of Application Procedural Interface (API) and Direct Interface (DI) functions that an application can invoke to interact with ACIS. API functions, which combine modeler functionality with application support features such as argument error checking and roll back, are the main interface between applications and ACIS. The DI functions provide access to modeler functionality, but do not provide the additional application support features, and, unlike APIs, are not guaranteed to remain consistent from release to release. Refer to the *3D ACIS Online Help User's Guide* for a description of the fields in the reference template.

api_bend_entity

Function: Space Warping

Action: Modifies an entity or list of entities by bending it around a specified axis.

Prototype:

```
outcome api_bend_entity (
    ENTITY* in_entity,      // entity to modify
    SPAposition& neutral_root,  // neutral plane
location
    SPAunit_vector&         // bending axis
        neutral_axis,       // bending axis
    SPAunit_vector&         // bending direction
        bend_direction,     // bending direction
    double radius,          // bend radius
    double angle,           // bend angle (in
                            // radians) = 0
    double width            // bend width = 0
        = -1,               // (radius*angle)
    logical f_center_bend   // bend plane remains
        = FALSE,            // fixed if TRUE
    int n_points            // num of positions
        = 0,
    SPAposition* bend_regions   // array of positions
        = NULL,             // on faces to be bent
    AcisOptions* opts = NULL// ACIS options
    );
```

Includes:

```
#include ″kernel/acis.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/data/entity.hxx″
#include ″baseutil/logical.h″
#include ″baseutil/vector/position.hxx″
#include ″baseutil/vector/unitvec.hxx″
#include ″warphusk/api/warp_api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description:

Neutral root, bending axis, and bending direction define a neutral plane for the bending operation. A neutral plane is the location where the material is not stretched or compressed during bending. The material above the neutral plane (along bending direction) is compressed and the material below the neutral plane is stretched. The location of a neutral plane varies with the type of material to be bent.

neutral_root is a position that defines the location of neutral plane.

bending_axis is a gvector that defines the rotational axis of bending action.

bending_direction is auxiliary SPAvector that is used to define a bending plane.

The bending axis, denoted by ba, and the bending direction, denoted by bd, define a bending plane with normal vector ba x bd. The bending axis and bending direction should be perpendicular to each other. The cross product of these two vectors defines the positive and negative sides of the entity to be bent.

bend_radius is a real number that specifies the radius to the neutral plane.

bend_angle is the angle in radians that specifies the amount to bend.

bend_width is the width of bend region. The bend_radius, bend_angle, and bend_width are used to determine the region to be bent. Because only two parameters are independent, bend_width can be optional. Assigning zero or negative values to any of the three parameters implies that the parameter is skipped. It is always desirable to use just two parameters among these three. However, if only the bend_radius is given, the entire entity is bent. These parameters are related by: bend_width=bend_radius*bend_angle.

f_center_bend (optional) is a Boolean type to specify final orientation, i.e., center bend or fix end bend. The f_center_bend is used to control the final orientation of the bend entity. If f_center_bend is set to TRUE, material on both sides of the bending plane are equally bent by half of the bend_angle. If the f_center_bend is not set or set to FALSE, the negative side is fixed, i.e., only the positive side is bent by bend_angle. This mode of bending is called fixed end bending.

bend_regions (optional) specifies the portions of the entity to bend. One or more positions can be provided. The positions must be either within the components to be bent, or on their faces. Several bend positions can be specified to localize bending operation. The given bend positions are tested to determine if they are within the bending region. The bend positions must lie on a face to be used. If any of the bend positions is invalid (outside the bending region or outside the entity and its faces), no bending is performed.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: Changes model

api_bend_to_curve_entity

Function: Space Warping

Action: Modifies an entity or list of entities by bending it using the same function that maps the specified line to the specified curve.

Prototype:

```
outcome api_bend_to_curve_entity (
    BODY* in_entity,         // entity to bend
    const SPAposition& start,   // initial line's
                             // start point
    const SPAposition& end,  // initial line's end
                             // point
    const SPAunit_vector&    // rail for initial
        initial_rail,        // line
                             // and final curve
    curve_law* final_curve,  // final curve
    law* final_rail = NULL,  // rail for final curve
    AcisOptions* opts = NULL // ACIS options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/vector/position.hxx″
#include ″baseutil/vector/unitvec.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/body.hxx″
#include ″lawutil/main_law.hxx″
#include ″warphusk/api/warp_api.hxx″
#include ″lawutil/law_base.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: Changes model

api_initialize_warp

Function: Space Warping, Modeler Control

Action: Initializes the space warping library.

Prototype:
```
outcome api_initialize_warp ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″warphusk/api/warp_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: System routine

api_simplify_body

Function: Space Warping

Action: Simplifies the faces and edges of a body.

Prototype:
```
outcome api_simplify_body (
    BODY* in_body,          // body to be simplified
    AcisOptions* ao = NULL  // ACIS options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/body.hxx″
#include ″warphusk/api/warp_api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: The faces and edges of an entity are replaced with analytic surfaces and curves, if possible. Otherwise, they are replaced with their B–spline approximations.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: Changes model

api_simplify_face

Function: Space Warping

Action: Simplifies the face and the edges of the face.

Prototype:
```
outcome api_simplify_face (
    FACE* in_face,          // face to be
                            // simplified
    logical reparam         // for future
        = FALSE,            // use
    AcisOptions* ao = NULL  // ACIS options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/face.hxx″
#include ″warphusk/api/warp_api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: The face and edges of the face are replaced with analytic surfaces and curves, if possible. Otherwise, they are replaced with their B–spline approximations.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: Changes model

api_space_warp

Function: Model Geometry, Laws, Space Warping

Action: Modifies a body based on the given input law.

Prototype:
```
outcome api_space_warp (
    BODY* in_body,          // body to warp
    law* in_law,            // law defining the warp
    AcisOptions* opts = NULL// ACIS options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”warphusk/api/warp_api.hxx”
#include ”lawutil/law_base.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: Changes model

api_stretch_entity

Function: Model Geometry, Laws, Space Warping

Action: Creates a stretch for a given region of a body along the given axis.

Prototype:

```
outcome api_stretch_entity (
    ENTITY* body,           // body to stretch
    SPAposition pos1,       // start position of
                            // stretch axis, defines
                            // plane1
    SPAposition pos2,       // end position of
                            // stretch axis, defines
                            // plane2
    double dis1,            // distance to offset
                            // points at plane 1
    double dis2,            // distance to offset
                            // points at plane 2
    int continuity,         // the continuity between
                            // stretched and
                            // unstretched sections
                            // (Only 0 is currently
                            // supported)
    AcisOptions* opts = NULL// ACIS options
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "baseutil/vector/position.hxx"
#include "warphusk/api/warp_api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description:

Entity stretching is a specialized space warping operation. The pos1 and pos2 represent endpoints of an axis along which stretching is to be performed and can be either inside or outside the body. Two planes are formed at the pos1 and pos2 positions which are perpendicular to the axis. They specify the region of the body where stretching is to occur.

The dis1 and dis2 parameters are real numbers that specify the translation of the non–stretched portions of the body. Typically, dis1 is 0, meaning that the portion of the body below the stretch's starting region retains its position. If a nonzero value for dis1 is specified the entire body is translated along the axis by the given distance before performing the stretch.

Let heightA be the distance between pos1 and pos2.

Let heightB be heightA + (dis2–dis1).

Let heightR be heightB/heightA.

heightR represents the amount of scaling to apply to the stretch region of the body, along the stretch axis. Thus, if heightR=1 then no stretching or scaling need be applied.

The continuity value refers the continuity between stretched and unstretched sections. Only 0 is currently supported.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: Changes model

api_terminate_warp

Function: Space Warping, Modeler Control

Action: Terminates the space warping library.

Prototype:
```
outcome api_terminate_warp ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”warphusk/api/warp_api.hxx”
#include ”kernel/kernapi/api/api.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: System routine

api_twist_entity

Function: Entity, Space Warping

Action: Creates a twist for a given region of a body about an axis by the specified amount.

Prototype:

```
outcome api_twist_entity (
    ENTITY* body,           // body to twist
    SPAposition& pos1,      // axis end point1
    SPAposition& pos2,      // axis end point2
    double theta1,          // rotation at start
    double theta2,          // rotation at end
    int continuity,         // G0, G1, or G2
    AcisOptions* opts = NULL// ACIS options
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "baseutil/vector/position.hxx"
#include "warphusk/api/warp_api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Entity twisting is a specialized space warping operation. The pos1 and pos2 represent endpoints of an axis about which twisting is to be performed and can be either inside or outside the body. Two planes are formed at the pos1 and pos2 positions which are normal to the axis. They specify the region of the body where twisting is to occur.

The theta1 and theta2 parameters are real numbers that specify in radians the orientation of the non –twisted portions of the body. Typically, theta1 is 0, meaning that the portion of the body below the twist's starting region retains its orientation to the coordinate system. If a nonzero value for theta1 is specified, the entire body is transformed about the axis by the given amount before performing the twist.

The difference between theta1 and theta2 represents the amount in radians that the twist region is warped around the axis. Both theta1 and theta2 can be greater than or multiples of +/– 2*pi degrees to provide more turns in the twisting region.

The continuity value can be 0, 1, or 2 and refer to G0, G1, and G2 continuity. The interpolation function uses a linear, cubic, or quintic polynomial to obtain G0, G1, or G2 continuity between twisted and untwisted sections.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/api/warp_api.hxx

Effect: Changes model

is_WARP_ANNOTATION

Function: Space Warping

Action: Determines if an ENTITY is a WARP_ANNOTATION.

Prototype:
```
logical is_WARP_ANNOTATION (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”kernel/kerndata/data/entity.hxx”
#include ”warphusk/warpanno.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: warphusk

Filename: warp/warphusk/warpanno.hxx

Effect: Read–only

is_WARP_ANNO_FACE

Function: Space Warping

Action: Determines if an ENTITY is a WARP_ANNO_FACE.

Prototype:
```
logical is_WARP_ANNO_FACE (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”kernel/kerndata/data/entity.hxx”
#include ”warphusk/warpanno.hxx”
```

Description: Refer to Action.

Errors: None

Limitations:	None
Library:	warphusk
Filename:	warp/warphusk/warpanno.hxx
Effect:	Read–only