Chapter 23.
Functions bs3_surface Sa thru Wz

Topic: Ignore

bs3_surface_same

Function: Spline Interface, Construction Geometry

Action: Determines whether two spline surfaces are the same.

Prototype:
```
logical bs3_surface_same (
    bs3_surface bs1,        // first surface
    bs3_surface bs2,        // second surface
    double tol              // tolerance of control
        = 0.0               // point positions
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Determines whether two spline surfaces are (apparently) the same. This is not a comprehensive test - surfaces that are coincident but have differing extents, knot vectors or other internal details will not be spotted. Use this routine as a filter.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Read–only

bs3_surface_save

Function: Spline Interface, Construction Geometry, SAT Save and Restore

Action: Saves a surface.

Prototype:

```
void bs3_surface_save (
    bs3_surface sur         // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Writes a representation of the parametric surface to some external medium, using routines write_int, write_long, write_real, and write_string, defined in ACIS file kernutil/fileio/fileio.hxx.

For the previous four routines, no single external format will be suitable for all possible parametric surface representations. But, where appropriate, use the definition for rational and nonrational B-splines for compatibility.

The overloaded << operator behaves like bs3_surface_save, except that it writes to a C++ style stream using stream operators. bs3_surface_save does not require the output format to be the same, but it is strongly recommended that it is. For example:

bs3_surface surf;

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: System routine

bs3_surface_set_closed_u

Function: Spline Interface, Construction Geometry

Action: Sets the bs3_surface to be closed in *u*.

Prototype:

```
void bs3_surface_set_closed_u (
    bs3_surface surf        // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description:	Refer to Action.
Errors:	None
Limitations:	None
Library:	kernel
Filename:	kern/kernel/spline/sg_bs3s/sps3srtn.hxx
Effect:	Changes model

bs3_surface_set_closed_v

Function: Spline Interface, Construction Geometry

Action: Sets the bs3_surface to be closed in *v*.

Prototype:
```
void bs3_surface_set_closed_v (
    bs3_surface surf         // given surface
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description:	Refer to Action.
Errors:	None
Limitations:	None
Library:	kernel
Filename:	kern/kernel/spline/sg_bs3s/sps3srtn.hxx
Effect:	Changes model

bs3_surface_set_ctrlpt

Function: Spline Interface, Construction Geometry

Action: Sets the position of one control point.

Prototype:
```
void bs3_surface_set_ctrlpt (
    bs3_surface surf,          // target bs3_surface to
                               // modify
    int uindex,                // u index of target
                               // control point
    int vindex,                // v index of target
                               // control point
    double* pos,               // xyz location copied
                               // into control
                               // point,size:[3]
    double weight              // weight to which
                               // control point is
                               // assigned only used if
                               // surf is rational
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Checks that surf has an (*i*,*j*) control point. If it does it copies the *xyz* values of pos into control point’s data structure. When surf is rational it also copies the weight value into the control point’s data structure.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_set_ctrlpts

Function: Spline Interface, Construction Geometry

Action: Sets the position of all control points.

Prototype:

```
void bs3_surface_set_ctrlpts (
    bs3_surface surf,          // target bs3_surface to
                               // modify
    int u_cpt_count,           // number of control
                               // points in u
    int v_cpt_count,           // number of control
                               // points in v
    double* pos_vec,           // control point
                               // locations [c00,
                               // c01, ... , c10, c11,
                               // ...] where cij = xyz
                               //for control point i,j.
                               // size = u_cpt_count *
                               // v_cpt_count * 3
    double* weight             // weight to which
                               // control points are
                               // assigned only used if
                               // surf is rational size
                               // = u_cpt_count *
                               // v_cpt_count
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Copies the new *xyz* values of the pos_vec array into surface’s control point data structure. If surf is rational, the weight values are also copied.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_set_form

Function: Spline Interface, Construction Geometry

Action: Sets the form of a bs3_surface.

Prototype:

```
void bs3_surface_set_form (
    bs3_surface                // given surface
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/sg_bs3s/sps3srtn.hxx"
```

Description: Valid forms are bs3_surf_periodic_ends, bs3_surf_closed_ends, bs3_surf_open_ends, or bs3_surf_unknown_ends.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sps3srtn.hxx

Effect: Changes model

bs3_surface_set_open_u

Function: Spline Interface, Construction Geometry

Action: Sets the bs3_surface to be open in *u*.

Prototype:
```
void bs3_surface_set_open_u (
    bs3_surface surf        // given surface
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/sg_bs3s/sps3srtn.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sps3srtn.hxx

Effect: Changes model

bs3_surface_set_open_v

Function: Spline Interface, Construction Geometry

Action: Sets the bs3_surface to be open in *v*.

Prototype:

```
void bs3_surface_set_open_v (
    bs3_surface surf        // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sps3srtn.hxx

Effect: Changes model

bs3_surface_set_periodic_u

Function: Spline Interface, Construction Geometry

Action: Marks the surface as being periodic in *u*.

Prototype:

```
void bs3_surface_set_periodic_u (
    bs3_surface surf        // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sps3srtn.hxx

Effect: Changes model

bs3_surface_set_periodic_v

Function: Spline Interface, Construction Geometry

Action: Marks the surface as being periodic in *v*.

Prototype:

```
void bs3_surface_set_periodic_v (
    bs3_surface surf        // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sps3srtn.hxx

Effect: Changes model

bs3_surface_shift_u

Function: Spline Interface, Construction Geometry

Action: Reparameterizes the given surface in *u*.

Prototype:

```
void bs3_surface_shift_u (
    double delta,           // parameter shift
                            // desired
    bs3_surface bs          // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Reparameterizes the given surface in place by adding the shift value to its *u* parameter values.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_shift_v

Function: Spline Interface, Construction Geometry

Action: Reparameterizes the given surface in *v*.

Prototype:
```
void bs3_surface_shift_v (
    double delta,            // parameter shift
                             // desired
    bs3_surface bs           // given surface
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Reparameterizes the given surface in place by adding the shift value to its *v* parameter values.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_sil

Function: Spline Interface, Construction Geometry

Action: Creates the silhouettes of the surface.

Prototype:
```
surf_surf_int* bs3_surface_sil (
    bs3_surface,             // given surface
    logical,                 // TRUE if surface is
                             // negated
    view_spec const&,        // view specification
    SPAbox const&            // region of interest
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/box.hxx”
#include ”kernel/kernint/intsfsf/sfsfint.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
#include ”intersct/kernint/makesil/makesil.hxx”
```

Description: Creates the silhouettes of the surface when viewed from the point or direction specified by the view_spec (defined in makesil/makesil.hxx).

The view_spec class contains the following information:

to_point is a position in object space, normally in the general neighborhood of the surface.

dir is the direction in object space from to_point to the view position, i.e., opposite to the view direction.

inv_dist is the reciprocal of the distance from to_point to the view position, along dir. This is zero for a parallel projection.

The results of this function are returned using the surf_surf_int class, but only some of the members are significant:

cur is a pointer to an ACIS curve that contains the object-space description of the intersection curve. In most cases this routine will construct an intcurve, but in special cases of parametric surfaces it may return straight lines or ellipses.

pcur1 is a pointer to parameter-space curves, specifying the curve in the parameter space of the surface.

start_term, start_param; If the curve starts at a branch-point (i.e., several silhouettes join there), start_term points to a surf_surf_term representing this phenomenon, and start_param gives its parameter value on this intersection curve. If not, start_term is NULL, and start_param undefined. Note that for a given point, all curve segments starting or ending there must point to the same surf_surf_term object, not to separate objects with the same position.

end_term, end_param; As for start_term, etc. but for the end of this curve.

nsplit, split_param; If this curve has more than one disjoint portion lying within the region of interest, split points must be provided which lie outside the region of interest and divide the curve into portions, each of which has at most one segment within the region of interest. These points are defined by an ordered sequence of parameter values.

left_surf_rel[0], right_surf_rel[0]; The visibility of the surface on either side of the curve. If the surface normal on the appropriate side of the curve points away from the view point (i.e., has a positive dot product with the view direction), the relationship is outside, otherwise inside. In this context, left and right are as viewed in the direction of the curve, with the surface normal upwards.

pcur2, left_surf_rel[1], right_surf_rel[1], int_type, aux_surf, and aux_left_rel[ ] are not used.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_singular_u

Function: Spline Interface, Construction Geometry

Action: Determines if the mapping from parameter space to object-space is singular along the given constant *u*-parameter line.

Prototype:

```
logical bs3_surface_singular_u (
    double u,                 // u parameter of
                              // interest
    bs3_surface bs            // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
 #include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: This routine returns TRUE if the parameter space to object space mapping is singular, otherwise it returns FALSE.

Normally, the only form of singularity allowed is where the whole parameter line maps to a single object-space point, and it may only occur at one end of the parameter range.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Read–only

bs3_surface_singular_v

Function: Spline Interface, Construction Geometry

Action: Determines if the mapping from parameter space to object space is singular along the given constant v parameter line.

Prototype:
```
logical bs3_surface_singular_v (
    double v,                // v parameter of
                             // interest
    bs3_surface bs           // given surface
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: If the mapping from parameter space to object space is singular along the given constant *v*-parameter line, return TRUE; otherwise, FALSE.

Normally, the only form of singularity allowed is where the whole parameter line maps to a single object space point, and it may only occur at one end of the parameter range.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Read–only

bs3_surface_skin_curves

Function: Spline Interface, Construction Geometry

Action: Creates a surface that interpolates (“skins”) between a sequence of bs3_curves.

Prototype:
```
bs3_surface bs3_surface_skin_curves (
    ENTITY_LIST& curves      // ordered list of
                             // bs3_curves
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: It does not handle coincident ends or first-order discontinuities.

Errors: If the curves cannot be made compatible, or if an error occurs, a NULL surface is returned.

Limitations: Rational curves may not be handled properly. Use this function for non–rational curves.

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sps3srtn.hxx

Effect: Changes model

bs3_surface_span_range_u

Function: Spline Interface, Construction Geometry

Action: Gets the *u* parameter bounds of a surface's simple patches.

Prototype:

```
SPAinterval bs3_surface_span_range_u (
    int i,                    // span number n in u
                              // direction
    bs3_surface sur           // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/interval.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: If the surface is an array of simple patches, numbered sequentially in the increasing *u* and *v* directions, starting at 0.0, return the *u* parameter bounds of the patches indexed in the *u* direction by the given integer.

This routine returns an empty interval if the index is out of the range implied by the value returned by bs3_surface_nspans_u.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Read–only

bs3_surface_span_range_v

Function: Spline Interface, Construction Geometry

Action: Gets the *v* parameter bounds of a surface’s simple patches.

Prototype:

```
SPAinterval bs3_surface_span_range_v (
    int j,                  // span number n in v
                            // direction
    bs3_surface sur         // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/interval.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: If the surface is an array of simple patches, numbered sequentially in the increasing *u* and *v* directions, starting at zero, this function returns the *v* parameter bounds of the patches indexed in the *v* direction by the given integer.

This routine returns an empty interval if the index is out of the range implied by the value returned by bs3_surface_nspans_v.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Read–only

bs3_surface_split_u

Function: Spline Interface, Construction Geometry

Action: Splits a B-spline surface into two sections at a given *u* parameter value.

Prototype:

```
bs3_surface bs3_surface_split_u (
    bs3_surface& sur,       // given surface
    double uparam           // given u parameter
                            // value
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: The part with *u* parameter less than that given is returned as the function value, and the supplied surface is modified to represent the remainder.

If the given parameter is at the beginning of the range, or before the beginning for a non–periodic surface, the function returns NULL, and leaves the original surface unchanged (except to mark a periodic surface as ”closed” instead).

Similarly, if the parameter is at the end of the range (or beyond for a non–periodic surface), the function returns the original surface, again unchanged except for ”closed” instead of ”periodic”.

For a periodic surface, the parameter range is first adjusted by a whole number of periods to bracket the given parameter.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_split_v

Function: Spline Interface, Construction Geometry

Action: Splits a B-spline surface into two sections at a given *v* parameter value.

Prototype:

```
bs3_surface bs3_surface_split_v (
    bs3_surface& sur,        // given surface
    double vparam            // given v parameter
value
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: The part with *v* parameter less than that given is returned as the function value, and the supplied surface is modified to represent the remainder.

If the given parameter is at the beginning of the range, or before the beginning for a non–periodic surface, the function returns NULL, and leaves the original surface unchanged (except to mark a periodic surface as ”closed” instead).

Similarly, if the parameter is at the end of the range (or beyond for a non–periodic surface), the function returns the original surface, again unchanged except for ”closed” instead of ”periodic”.

For a periodic surface, the parameter range is first adjusted by a whole number of periods to bracket the given parameter.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_subset

Function: Spline Interface, Construction Geometry

Action: Creates a surface identical to the given surface.

Prototype:

```
bs3_surface bs3_surface_subset (
    bs3_surface old_bs,      // given surface
    SPApar_box const&        // required boundary
        new_range,           // range
    double                   // required positional
         = 0,                // fit
    double& actual_fit       // actual fit used
        =*(double*)NULL_REF
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/param.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Creates a surface identical to the given surface (including parameterization) within the overlap between the given parameter range and the range of the surface (taking into account periodicity), but not necessarily defined outside.

If the subset cannot be taken, or is deemed not to be worthwhile, the routine may return NULL. The routine may return a surface that is only a fit to the true subset surface, to the specified precision. Because most systems will represent such a surface exactly, it is unlikely that this option will be exercised. In this case, the actual fit value will be returned as exact 0.

Errors: Returns NULL if the input surface is NULL.

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_sum_curves

Function: Spline Interface, Construction Geometry

Action: Creates a spline surface by summing two 3D B-spline curves.

Prototype:

```
bs3_surface bs3_surface_sum_curves (
    bs3_curve u_bs,          // curve in u direction
    bs3_curve v_bs,          // curve in v direction
    double                   // requested fit
         = 0,                // tolerance
    double& actual_fit       // returned actual fit
        =*(double*)NULL_REF  // tolerance used
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_crv/bs3curve.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Constructs a spline surface from two spline curves, as the simple sum. The second curve is translated so that its start point sweeps along the first curve, and the surface is what gets swept out. The resulting *u* parameter curves are all translations of the first curve, *v*–parameter curves are translations of the second.

The start points of the two curves need not match, in which case the first curve is used to define the low *v* parameter line of the surface. All other parameter lines are translations of the given curves. It is unlikely that the precision arguments will ever be need, but they are included here for consistency.

If the *u*-direction curve or the *v*-direction curve are closed or periodic, the resulting surface is closed or periodic in that parameter. Both curves must *not* be closed or periodic, because the resulting surface would be self-intersecting.

Errors:	None
Limitations:	None
Library:	kernel
Filename:	kern/kernel/spline/bs3_srf/sp3srtn.hxx
Effect:	Changes model

bs3_surface_sum_x_pla_str

Function: Spline Interface, Construction Geometry

Action: Specializes the top curve construction for perpendicular sweep.

Prototype:

```
bs3_curve bs3_surface_sum_x_pla_str (
    const bs3_surface in_sur,    // given sum surface
    const SPAposition&,          // path start
    const SPAunit_vector& path_dir,// path direction
    const SPAposition& plane_root, // miter plane
                                 // root point
    const SPAunit_vector& plane_nor // miter plane
                                 // normal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”kernel/spline/bs3_crv/bs3curve.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: Specializes the top curve construction for perpendicular sweep, which results in an exact sum surface as the lateral_surface. The output is the projection of the low_u curve of the sum surface on to the miter plane along the given straight path.

Errors:	None
Limitations:	None
Library:	kernel
Filename:	kern/kernel/spline/sg_bs3s/sps3srtn.hxx
Effect:	Changes model

bs3_surface_testpt

Function: Spline Interface, Construction Geometry

Action: Determines whether an object-space point lies on a given surface within the given positional precision.

Prototype:
```
logical bs3_surface_testpt (
    SPAposition const& pos, // given point
    double tol,             // permitted tolerance
    bs3_surface bs,         // given surface
    SPApar_pos const& uv_guess// approximation to the
        =*(SPApar_pos*)NULL_REF, // parameter value of
                            // the foot of the
                            // perpendicular from the
                            // point to the surface
    SPApar_pos& uv_actual   // returned actual
        =*(SPApar_pos*)NULL_REF // parameter value
                            // used
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/param.hxx”
#include ”baseutil/vector/position.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: This function takes advantage of an estimate of the surface parameter values to speed up processing, but works correctly without them.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Read–only

bs3_surface_to_array

Function: Spline Interface, Construction Geometry

Action: Creates arrays of control points, weights, *u* knots, and *v* knots from a B–spline surface.

Prototype:

```
void bs3_surface_to_array (
    bs3_surface srf,          // given surface
    int& dim,                 // returned dimension
    logical& rational_u,      // returned rational in u
    logical& rational_v,      // returned rational in v
    int& form_u,              // returned form in u
    int& form_v,              // returned form in v
    int& pole_u,              // returned poles in u
    int& pole_v,              // returned poles in v
    int& num_u,               // returned number of
                              // control points in u
    int& num_v,               // returned number of
                              // control points in v
    SPAposition*& ctrlpts,    // returned control
                              // points in desired
                              // order
    double*& weights,         // returned weights
    int& degree_u,            // returned degree in u
    int& num_uknots,          // returned number of
                              // knots in u
    double*& uknots,          // returned knots in u
    int& degree_v,            // returned degree in v
    int& num_vknots,          // returned number of
                              // knots in v
    double*& vknots           // returned knots in v
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/position.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: The surface is defined by an array of control points, weights, and knots in the *u* *(v)* parameter. The surface may be rational in either *u* or *v*, it may be open, closed, or periodic in *u* or *v*, and it may have parametric singularities at the minimum or maximum parameter values in either *u* or *v*.

The control points are returned as an array of coordinates in the form (x,y,z) or (x,y,z).

The function creates arrays of control points, weights, *u* knots, and *v* knots. It is up to the application to delete these arrays.

rational_u and rational_v specify if surface is rational (1) or not rational (0).

form_u and form_v specify if the surface is open (0), closed (1), or periodic (2).

pole_u and pole_v specify if the surface has poles at none (0), low *u*/*v* (1), high *u*/*v*, or both.

If the surface has multiple end knots, the knot arrays returned have same knots at start and end up to respective degrees + 1.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sps3srtn.hxx

Effect: Changes model

bs3_surface_trans

Function: Spline Interface, Construction Geometry, Transforms, Modifying Models

Action: Transforms a surface.

Prototype:

```
void bs3_surface_trans (
    bs3_surface sur,         // given surface
    SPAtransf const& t       // transform
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/transf.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: A transform consists of a 3 x 3 matrix with unit determinant, giving an affine transformation, an overall scaling factor, and a translation vector. There are three logical flags, relating to the matrix.

Rotate indicates whether the matrix is anything other than the identity.

Reflect indicates whether the determinant is -1.

Shear is set if the matrix isn’t orthogonal.

The parameterization of a surface must be independent of transformation. Therefore, the result of evaluating a transformed surface will be the same as evaluating the untransformed surface at the same parameter value, and transforming the result.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Changes model

bs3_surface_unitvec

Function: Spline Interface, Construction Geometry

Action: Determines the offset in parameter space corresponding to a unit move in a direction on a 3D B-spline surface at a given position.

Prototype:

```
SPApar_vec bs3_surface_unitvec (
    SPAunit_vector const& dir,// given direction
    SPApar_pos const& uv,    // given parameter point
    bs3_surface bs           // given surface
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/param.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/sp3srtn.hxx”
```

Description: Refer to Action.

Errors: Returns an empty SPApar_vec if the input surface is null.

Limitations: None

Library: kernel

Filename: kern/kernel/spline/bs3_srf/sp3srtn.hxx

Effect: Read–only

bs3_surface_weights

Function: Spline Interface, Construction Geometry

Action: Gets the number of weights in the u and v directions and the array of weights for the given surface.

Prototype:

```
void bs3_surface_weights (
    bs3_surface bs,          // input surface
    int& num_u,              // number of weights in u
    int& num_v,              // number of weights in v
    double*& weights         // array of weights
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: This function creates an array of weights for the given surface. The length of the array is num_u*num_v. The order the weights are stored in the array is [u][v], such that v increments more quickly. It is the responsibility of the calling application to delete the weights array.

Errors: None.

Limitations: None

Library: kernel

Filename: kern/kernel/spline/sg_bs3s/sp3srtn.hxx

Effect: Read–only

bs3_surface_wiggle

Function: Spline Interface, Construction Geometry

Action: Creates four splines for the edges or a wiggle.

Prototype:

```
bs3_surface bs3_surface_wiggle (
    double width,            // width
    double depth,            // depth
    double height,           // height
    int low_v_type           // low v type
        = 1,
    int high_v_type          // high v type
        = -2,
    int low_u_type           // low u type
        = 2,
    int high_u_type          // high u type
        = -1
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/bs3_srf/cc_rout.hxx”
```

Description:	Creates four splines for the edges. Each is a cubic B-spline with two spans, passing through three colinear points. The shape is specified by the appropriate integer argument, as follows:

0	= straight line
1	= S shape, starting at low parameter value with a 45 degree upward (positive z) tangent, and ending at high parameter in the same direction.
2	= double hump, starting going upwards and ending downwards.
–1	= same as 1, but inverted.
.–2	= same as 2, but inverted.

Errors:	None.
Limitations:	None
Library:	kernel
Filename:	kern/kernel/spline/bs3_srf/cc_rout.hxx
Effect:	Changes model