Chapter 21.
Functions bs3_surface Aa thru Lz

Topic: Ignore

bs3_surface_3crv

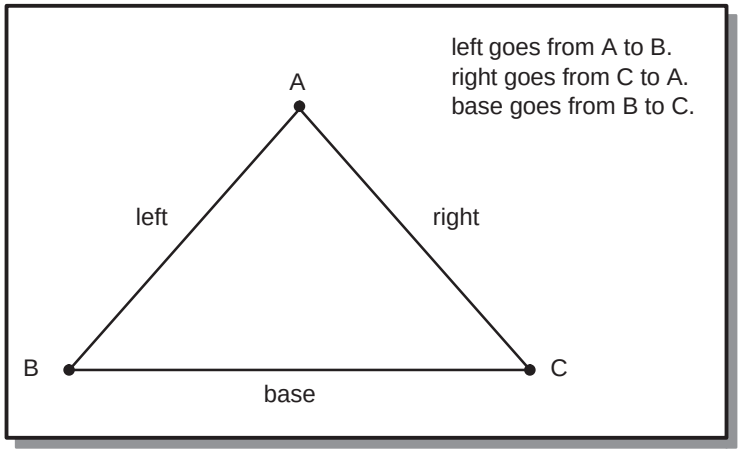
Function: Spline Interface, Construction Geometry

Action: Creates a spline surface that interpolates three boundary curves.

Prototype:
```
bs3_surface bs3_surface_3crv (
    const bs3_curve& base,  // edge defining curve
    const bs3_curve& right, // edge defining curve
    const bs3_curve& left   // edge defining curve
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_crv/bs3curve.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: The end points of the curves and their directions match up as shown in the figure. The base defines the *u* parameterization and the left and right sides define the *v*. The apex *A* is a singularity.



Errors: If an error occurs, a NULL surface is returned. The original curves remain.

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_4crv

Function: Spline Interface, Construction Geometry

Action: Creates a spline surface that interpolates four boundary curves.

Prototype:
```
bs3_surface bs3_surface_4crv (
    const bs3_curve& bottom, // edge defining curve
    const bs3_curve& right,  // edge defining curve
    const bs3_curve& top,    // edge defining curve
    const bs3_curve& left    // edge defining curve
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_crv/bs3curve.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/sg_bs3s/sps3srtn.hxx”
```

Description: The correspondence of the curves and the surface patch is:

```
bottom--------> S (u,v0)   (oriented left to right)
top-----------> S (u,v1)   (oriented left to right)
left----------> S (u0,v)   (oriented bottom to top)
right---------> S (u1,v)   (oriented bottom to top)
```

Errors: If an error occurs, a NULL surface is returned, and the original curves remain.

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_accurate_derivs

Function: Spline Interface, Construction Geometry

Action: Gets the number of derivatives that bs3_surface_evaluate evaluates accurately.

Prototype:
```
int bs3_surface_accurate_derivs (
    bs3_surface             // given surface
    );
```

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

Description: Returns the number of derivatives that bs3_surface_evaluate evaluates accurately.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_add_knot

Function: Spline Interface, Construction Geometry

Action: Adds knots to a surface.

Prototype:
```
int bs3_surface_add_knot (
    double par,             // given parameter
    int multp,              // multiplicity wanted
                            // at added knot
    bs3_surface in_sur,     // given surface
    int u_or_v,             // add a knot: 0=u,
                            // 1=v direction
    double knot_tol         // knot tolerance
    );
```

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

Description: Adds knot to a surface up to the requested multiplicity with in the given knot tolerance. This routine returns the number of knots added.

The input parameter has to be with in the parameter bounds of the given surface. The final multiplicity of the added knot cannot be greater than the degree of the surface in the requested direction.

Use the tolerance to distinguish between the knots. It is used to test whether the knot being added already exists.

Errors: None

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_bicubic

Function: Spline Interface, Construction Geometry

Action: Creates a surface that is a bi-cubic interpolant.

Prototype:
```
bs3_surface bs3_surface_bicubic (
    int num_upts,           // number of knots in u
    int num_vpts,           // number of knots in v
    double u_params[],      // u knots, size num_upts
    double v_params[],      // v knots, size num_vpts
    SPAposition points[],   // points on surface
                            // implicit 2D array
    SPAvector u_tans[],     // tangent vectors in u
                            // direction – implicit
                            // 2D array
    SPAvector v_tans[],     // tangent vectors in v
                            // direction – implicit
                            // 2D array
    SPAvector twists[]      // twist vectors –
                            // implicit 2D array
    );
```

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

Description: This routine interpolates a mesh of points, *u* tangent, *v* tangents and twist vectors at each point. A bs3_surface is constructed and returned as the function return. The routine also requires knot values associated with each point that is interpolated in both *u* and *v* direction.

The control points are contained in an array of positions. The *v* index varies first. That is, a row of *v* control points for the first *u* value is specified first. Then, the row of *v* control points for the next *u* value. The other 2D arrays are specified in the same order.

The size of the points, u_trans, v_trans, and twists arrays is num_upts*num_vpts, and the ordering is [num_upts][num_vpts].

Errors: If an error occurs, the function returns a NULL surface.

Limitations: No two adjacent points to be interpolated can be same within tolerance. However the interpolated points can be same at start and end for closed surfaces, so does other types of configurations are allowed which result in self–intersecting surfaces.

Library: kernel

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

Effect: Changes model

bs3_surface_bispan

Function: Spline Interface, Construction Geometry

Action: Creates a surface from a simple patch of a surface.

Prototype:
```
bs3_surface bs3_surface_bispan (
    int i,                  // span number in u
                            // direction
    int j,                  // span number in v
                            // direction
    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: Creates a new surface that consists only of the *(i,j)*th simple patch of the given surface. The knot vectors of the new surface will have start and end multiplicities equal to the degree; therefore, the new surface will represent a single Bezier patch. In the case of a rational surface, the weights associated with the boundary control points have not been normalized.

Errors: If an error occurs, this function returns a NULL surface.

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_bispan_poly

Function: Spline Interface, Construction Geometry

Action: Converts a span into a rational bipolynomial vector with normalized parameterization in each direction.

Prototype:
```
rat_bipoly_vec bs3_surface_bispan_poly (
    int nuspan,             // ith span in u
                            // direction
    int nvspan,             // ith span in v
                            // direction
    bs3_surface sur         // given surface
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kernutil/bipoly/bipoly.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/bs3_srf/sp3srtn.hxx"
```

Description: Converts the (*i*,*j*)th span into a rational bipolynomial vector, with a normalized [0, 1] parameterization in each direction. Assumes that a bs3_surface is a piecewise rational bipolynomial vector function of its parameters. The return type is rat_bipoly_vec.

Errors: If an error occurs, this routine returns the zero vector.

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_bispan_range

Function: Spline Interface, Construction Geometry

Action: Gets the parameter bounds of a simple surface patch.

Prototype:
```
SPApar_box bs3_surface_bispan_range (
    int i,                  // span number in u
    int j,                  // span number in v
    bs3_surface sur         // given surface
    );
```

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: Returns the parameter bounds of the *(i,j)*th simple patch of the surface, where the argument i runs from 0 for the first span to one fewer than the number returned by bs3_surface_nspans_u, and j is similar in the *v*-direction.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_boundary_angle

Function: Spline Interface, Construction Geometry

Action: Gets the boundary angle.

Prototype:
```
void bs3_surface_boundary_angle (
    bs3_surface sur,        // given surface
    double& u_angle,        // u angle
    double& v_angle         // v angle
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/d3_bs3/spd3rtn.hxx"
```

Description: Returns the maximum turning angle over the surface boundaries.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/d3_bs3/spd3rtn.hxx

Effect: Read–only

bs3_surface_box

Function: Spline Interface, Construction Geometry

Action: Gets a box that encloses a portion of a three-dimensional B-spline surface.

Prototype:
```
SPAbox bs3_surface_box (
    bs3_surface bs,          // given surface
    SPApar_box const&        // parameter range of
          =*(SPApar_box*)NULL_REF// interest
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/box.hxx"
#include "baseutil/vector/param.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/bs3_srf/sp3srtn.hxx"
```

Description: The box will not be the smallest possible, but will be a compromise between a tight fit and fast evaluation.

If the parameter box is NULL, the box will contain the whole surface. ACIS ensures that any parameter box given is entirely within the parameter range for the surface; however, if the surface is periodic, it may be partially or wholly outside the basic range of the periodic parameter.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_check

Function: Spline Interface, Construction Geometry

Action: Checks for errors in the approximating surface.

Prototype:
```
check_status_list* bs3_surface_check (
    bs3_surface bs3,        // given surface
    const spline& spl       // given spline
        =*(spline*)NULL_REF,
    const check_fix& fix    // available fixes
        =*(check_fix*)NULL_REF,
    check_fix& fixed        // fixes made
        =*(check_fix*)NULL_REF,
    const check_status_list*// list of things
        check               // to be checked
        = NULL
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "kernel/kerngeom/surface/spldef.hxx"
#include "kernel/kernint/d3_chk/chk_stat.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/d3_bs3/spd3rtn.hxx"
```

Description: If supplied with a spline, this extension uses that for evaluation in the continuity check.

In addition, it checks whether the control points of a bs3 surface are valid, whether for coincident adjacent control points, and surfaces which are closed but shouldn't be, or surfaces which are not closed but should be.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/d3_bs3/spd3rtn.hxx

Effect: Read–only

bs3_surface_closed_u

Function: Spline Interface, Construction Geometry

Action: Determines whether a given surface is closed in the *u*-parameter.

Prototype:
```
logical bs3_surface_closed_u (
    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 line on the surface corresponding to minimum *u*-parameter is geometrically identical to that for maximum *u*-parameter; otherwise it returns FALSE.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_closed_v

Function: Spline Interface, Construction Geometry

Action: Determines whether the given surface is closed in the *v*-parameter.

Prototype:
```
logical bs3_surface_closed_v (
    bs3_surface bs          // given surface
    );
```

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

Description: This routine returns TRUE if the parameter line on the surface corresponding to minimum *v*-parameter is geometrically identical to that for maximum *v*-parameter; otherwise it returns FALSE.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_control_points

Function: Spline Interface, Construction Geometry

Action: Gets the number of control points in the *u* and *v* directions, and the array of control points, for the given surface.

Prototype:
```
void bs3_surface_control_points (
    bs3_surface bs,          // input surface
    int& num_u,              // number of u control
                             // points
    int& num_v,              // number of v control
                             // points
    SPAposition*& ctrlpts    // control points array
    );
```

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

Description: This function creates an array of the control points for the given surface. The length of the array is num_u*num_v. The order the control points 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 control point array.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_copy

Function: Spline Interface, Construction Geometry

Action: Creates an exact copy of the surface in free store.

Prototype:
```
bs3_surface bs3_surface_copy (
    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: ACIS calls this routine only when a change is to be made to one copy of the surface, so there is no advantage to be gained by deferring the duplication further. Ordinary duplication of ACIS spline surfaces merely creates a new reference to the same underlying bs3_surface.

Errors: None

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_cross

Function: Spline Interface, Construction Geometry

Action: Evaluates the cross curvature of a three-dimensional B-spline surface at a given *uv*.

Prototype:
```
double bs3_surface_cross (
    SPApar_pos const& uv,    // given parameter point
    SPAunit_vector const& dir,// object space tangent
                              // direction
    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: Evaluates the cross curvature of a three-dimensional B-spline surface at a given *uv*.

This is equivalent to constructing a plane through the given point and perpendicular to the given direction, which must be tangent to the surface at the given point, and returning the curvature of the intersection curve between the plane and the surface at that point.

If the intersection curve is convex when viewed from the outside of the surface (the side that the normal points toward), the sign of the result is negative. If the curve is is concave, the sign is positive. If there is a discontinuity in curvature at the given point, the value returned is for the left-hand side of the intersection curve as viewed in the given tangent direction with the surface normal upwards.

Errors: Returns –1 if the input surface is null.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_debug

Function: Spline Interface, Construction Geometry, Debugging

Action: Gets a readable representation of the curve and writes it to a file.

Prototype:
```
void bs3_surface_debug (
    bs3_surface sur,         // given surface
    char const* leader,      // string to precede
                             // second and subsequent
                             // lines
    FILE* fp                 // output file
        = debug_file_ptr
    );
```

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

Description: Produces a readable representation of the surface on the given open file, in any convenient format.

If this extends to more than one text line (as is almost certain), start all lines but the first with the leader string. Do not terminate the last line by a newline character.

If the intersection curve is convex when viewed from the outside of the surface (the side that the normal points toward), the sign of the result is negative. If the curve is is concave, the sign is positive. If there is a discontinuity in curvature at the given point, the value returned is for the left-hand side of the intersection curve as viewed in the given tangent direction with the surface normal upwards.

Errors: none

Limitations: None

Library: kernel

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

Effect: System routine

bs3_surface_degree_u

Function: Spline Interface, Construction Geometry

Action: Gets the spline degree in the u direction.

Prototype:
```
int bs3_surface_degree_u (
    bs3_surface bs          // 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: Returns –1 if the input surface is null.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_degree_v

Function: Spline Interface, Construction Geometry

Action: Gets the spline degree in the v direction.

Prototype:
```
int bs3_surface_degree_v (
    bs3_surface bs          // 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: Returns –1 if the input surface is null.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_delete

Function: Spline Interface, Construction Geometry

Action: Deletes storage occupied by the given surface.

Prototype:
```
void bs3_surface_delete (
    bs3_surface& bs          // given surface
    );
```

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

Description: No assumptions are made by ACIS about how the underlying surface package manages its storage space, provided that it does not prevent the standard C memory allocation mechanism from working.

Errors: None.

Limitations: None

Library: kernel

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

Effect: System routine

bs3_surface_dim

Function: Spline Interface, Construction Geometry

Action: Gets the dimensionality of a surface.

Prototype:
```
int bs3_surface_dim (
    bs3_surface bs           // input surface
    );
```

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

Description: Returns the dimensionality of the surface. Usually this will be 3.

Errors: None.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_dim

Function: Spline Interface, Construction Geometry

Action: Gets the dimensionality of a surface.

Prototype:
```
int bs3_surface_dim (
    bs3_surface bs          // input surface
    );
```

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

Description: Returns the dimensionality of the surface. Usually this will be 3.

Errors: None.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_estimate_param

Function: Spline Interface, Construction Geometry

Action: Estimates the parameter values of the foot of a perpendicular from a given point to the surface.

Prototype:
```
SPApar_pos bs3_surface_estimate_param (
    SPAposition const& pos, // given point
    bs3_surface surface,    // given surface
    logical recurse         // use recursion
        = FALSE
    );
```

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: It is expected that this function, followed by a call to bs3_surface_perp using the estimated parameter value, will be substantially faster than a call to bs3_surface_perp with no estimated parameter value; however, the result may not give the nearest perpendicular, even if the given point is very near to the surface. Use bs3_surface_estimate_param only with algorithms that are resistant to such unexpected results.

Errors: None.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_eval

Function: Spline Interface, Construction Geometry

Action: Evaluates a bs3_surface for position, first, and second derivatives at the given parameter value.

Prototype:
```
void bs3_surface_eval (
    SPApar_pos const& uv,    // given parameter point
                             // uv
    bs3_surface bs,          // given surface
    SPAposition& pos,        // position returned
    SPAvector* d1            // du and dv returned in
        = NULL,              // an array of 2 vectors
    SPAvector* d2            // duu, duv and dvv
        = NULL               // returned in array
                             // of 3 vectors
    );
```

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: If pos is a non-null reference, it is set to the evaluated position of the surface at the given parameter values. If d1uv is not NULL, it must point to an array of vectors of length 2, and these are set to the surface derivatives with respect to the parameters u and v respectively. If d2uv is not NULL, it must point to an array of vectors of length 3, and these are set to the second derivatives of the surface, with respect to u twice, u and v, and v twice. (For all ordinary surfaces we may assume that the derivative with respect to v and u will be the same as that with respect to u and v.)

Errors: None.

Limitations: There is no provision to handle discontinuities of second derivative, so it is assumed that the second derivatives are continuous across the portion of the surface that is of interest. Also, it is assumed that the first derivatives are continuous everywhere. The direction of the surface normal is always required to be continuous throughout the interior of the portion of interest.

Library: kernel

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

Effect: Read–only

bs3_surface_evaluate

Function: Spline Interface, Construction Geometry

Action: Evaluates the position and an arbitrary number of derivatives of the surface.

Prototype:

```
int bs3_surface_evaluate (
    SPApar_pos const& uv,    // given parameter
                             // position
    bs3_surface sur,         // given surface
    SPAposition& pos,        // returned position
    SPAvector** deriv        // returned derivatives
        = NULL,              // array of pointers to
                             // arrays of vectors,each
                             // such array containing
                             // one more vector than
                             // the order of the
                             // derivative
    int nderiv               // returned number of
        = 0,                 // derivatives
                             // to be evaluated, equal
                             // to the length of the
                             // deriv array
    int uindex               // returned senses at u
        = 0,                 // discontinuities
    int vindex               // returned senses at v
        = 0                  // discontinuities
    );
```

Includes:

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

Description: Evaluates the position and an arbitrary number of derivatives of the surface, with control over the handedness of the evaluation if the derivatives are discontinuous.

deriv specifies an array of pointers to arrays, containing at least nderiv values, though any or all may NULL. If not NULL, entry n (representing the (n+1)th derivative) must point to an array of at least n+1 vectors.

uindex specifies the sense of evaluation at a u discontinuity: negative means evaluate to the left, positive means evaluate to the right, and 0 means “don’t care.”

vindex specifies the sense of evaluation at a v discontinuity: negative means evaluate below, positive means evaluate above, and 0 means “don’t care.”

Errors: Returns –1 if input surface is null, or nderiv < 0.

Limitations: There is a limit on the number of evaluated derivatives, as returned by bs3_surface_accurate_derivs; any further derivatives requested are set to 0.

Library: kernel

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

Effect: Read–only

bs3_surface_fit

Function: Spline Interface, Construction Geometry

Action: Fits a mesh of points to a bs3_surface.

Prototype:
```
bs3_surface bs3_surface_fit (
    double fittol,          // fit tolerance
    int num_u,              // number of points in u
    int num_v,              // number of points in v
    const SPAposition pts[],// points
    const SPAunit_vector    // u derivatives
        du_s[],             // at start
    const SPAunit_vector du_e[]// u derivatives at
end
    );
```

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

Description: Fits a mesh of points using optionally specified start and end derivatives in the u direction only. The start and end derivatives must all be specified or all be NULL.

The control points are contained in an array of positions. The v index varies first. That is, a row of v control points for the first u value is specified first. Then, the row of v control points for the next u value. The (num_v) u tangent vectors are specified in increasing v order.

Errors: If an error occurs, this routine returns a NULL surface.

Limitations: The distance between any two adjacent points to be fitted can not be within tolerance. However the fit points can be the same at the start and end for closed surfaces, and similarly for other types of configurations which result in self–intersecting surfaces.

Library: kernel

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

Effect: Changes model

bs3_surface_fitol

Function: Spline Interface, Construction Geometry

Action: Determines the fit tolerance of a surface.

Prototype:
```
double bs3_surface_fitol (
    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: The computed fit tolerance is not less than 10*SPAresabs.

Errors: None.

Limitations: None.

Library: kernel

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

Effect: Read–only

bs3_surface_from_ctrlpts

Function: Spline Interface, Construction Geometry

Action: Creates a B-spline surface from a collection of control points and knot vectors.

Prototype:

```
bs3_surface bs3_surface_from_ctrlpts (
    int degree_u,            // degree in u
    logical rational_u,      // rational in u
    int form_u,              // type in u
    int& pole_u,             // pole in u
    int num_ctrlpts_u,       // number of control
                             // points in u
    int degree_v,            // degree in v
    logical rational_v,      // rational in v
    int form_v,              // type in v
    int& pole_v,             // pole in v
    int num_ctrlpts_v,       // number of control
                             // points in v
    const SPAposition        // position control
        ctrlpts[],           // points
    const double weights[],  // weights
    double,                  // tolerance to determine
                             // if two control points
                             // are the same
    int num_knots_u,         // number of knots in u
    const double knots_u[],  // knots in u
    int num_knots_v,         // number of knots in v
    const double knots_v[],  // knots in v
    double knot_tol          // tolerance to determine
                             // if two knots are the
                             // same
    );
```

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 spline is defined by the given sequence of control points and knots. If the argument rational_u is TRUE, the surface is rational in the u parameter; if FALSE, it is not. Similarly, for rational_v.

The argument form_u specifies whether the surface is open (0), closed (1), or periodic (2) in the u direction. Similarly for form_v.

The argument pole_u indicates whether or not the surface has a singularity at the u-min or u-max parameter boundaries according to the following:

0 = No singularity at u-min or u-max boundary

1 = Has a singularity at the u-min boundary

2 = Has a singularity at the u-max boundary

3 = Has a singularity at both boundaries

Similarly for pole_v.

The control points are contained in an array of positions. The v index varies first. That is, a row of v control points for the first u value is found first. Then, the row of v control points for the next u value. If the surface is rational in either parameter, it is considered a rational surface and the associated weights are in the array of doubles. The values in this array are in the same sequential order as the control points.

The point_tol tolerance value determines when two control points are identical and the knot_tol tolerance value performs the same function for the knot sequence.

Errors: None.

Limitations: The knots input have to be in an strictly increasing order.

Library: kernel

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

Effect: Changes model

bs3_surface_hermite

Function: Spline Interface, Construction Geometry

Action: Creates a single patch Bezier surface from Hermite data at the patch corners.

Prototype:

```
bs3_surface bs3_surface_hermite (
    const SPAposition* corners, // corners
    const SPAvector* uderivs,   // u derivatives
    const SPAvector* vderivs,   // v derivatives
    const SPAvector* twists     // twists
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/vector.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/d3_bs3/spd3rtn.hxx"
```

Description: The only point to bear in mind is that data is passed in ascending u order first, but ag_srf_data thinks it's in ascending v order first. Therefore care must be taken when copying control points into the big array.

The arrays each have length 4, and contain data at (0,0), (1,0), (0,1) and (1,1) in that order. The derivatives are with respect to a unit parameterization.

Errors: None.

Limitations: None.

Library: kernel

Filename: kern/kernel/spline/d3_bs3/sp3srtn.hxx

Effect: Changes model

bs3_surface_hermite_intp

Function: Spline Interface, Construction Geometry

Action: Creates a bi-cubic Hermite interpolant using a mesh of points, tangents, twists, and knot vectors.

Prototype:
```
bs3_surface bs3_surface_hermite_intp (
    int nu,                 // number of points in u
    int nv,                 // number of points in v
    SPAposition* pts,       // object space points
                            // array [nu][nv]
    SPAvector* u_partials,  // u partial array
                            // [nu][nv]
    SPAvector* v_partials,  // v partial array
                            // [nu][nv]
    SPAvector* uv_partials, // uv partial array
                            // [nu][nv]
    double* u_knots,        // u knots array [nu]
    double* v_knots         // v knots array [nv]
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/vector/position.hxx″
#include ″baseutil/vector/vector.hxx″
#include ″kernel/spline/bs3_srf/bs3surf.hxx″
#include ″kernel/spline/d3_bs3/spd3rtn.hxx″
```

Description: Creates a bi-cubic Hermite interpolant using a mesh of points, u tangents, v tangents, uv twists, and the corresponding knot vectors.

Errors: If an error occurs, the function returns a NULL surface.

Limitations: No two adjacent points to be interpolated can be same within tolerance. However the interpolated points can be same at start and end for closed surfaces, so other types of configurations are allowed which result in self–intersecting surfaces.

Library: kernel

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

Effect: Changes model

bs3_surface_hull_planes

Function: Spline Interface, Construction Geometry

Action: Creates a bounding hull around a surface.

Prototype:
```
int bs3_surface_hull_planes (
    bs3_surface surface_,    // given surface
    SPAposition* points,     // six points on planes
    SPAunit_vector* normals // six plane normals
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/vector/position.hxx″
#include ″baseutil/vector/unitvec.hxx″
#include ″kernel/spline/bs3_srf/bs3surf.hxx″
#include ″kernel/spline/bs3_srf/sp3srtn.hxx″
```

Description: The hull is a set of six planes that (in some sense) closely bounds the entire surface. In principle, these could be the standard box planes, but this function will probably generate a significantly tighter bound, preferably transformation-independent, at the expense of moderate extra effort. By choosing two planes to be roughly parallel to the surface, the distance between them will fall as the square of the linear dimension of a smooth surface. As that dimension gets small, the volume of the bound can be made to fall more rapidly than the cube of the length.

Errors: None.

Limitations: Call this function only after a box test using the standard boxes. It returns the number of planes constructed and allocates an array of planes in free space, which it is the caller's responsibility to delete.

Library: kernel

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

Effect: Changes model

bs3_surface_ij_ctrlpt

Function: Spline Interface, Construction Geometry

Action: Gets the [*i*,*j*] control point of a given spline surface.

Prototype:
```
void bs3_surface_ij_ctrlpt (
    bs3_surface in_sur,      // given surface
    int i,                   // ith points in u
                             // direction
    int j,                   // jth points in v
                             // direction
    SPAposition& ctrl_pos,   // returned object space
                             // control point
    double& weight,          // weight if is_rational
    logical& is_rational,    // set TRUE if surface is
                             // rational in u and–or v
    int& dimension           // dimension of the
                             // object space point,
                             // >3 is not supported
    );
```

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: Refer to Action.

Errors: Returns (–1,–1,–1) on bad input: null surface or negative indices.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_ij_knu

Function: Spline Interface, Construction Geometry

Action: Gets the *i*th knot in *u* direction.

Prototype:
```
double bs3_surface_ij_knu (
    bs3_surface bs,          // given surface
    int i,                   // ith point in u
                             // direction
    int j                    // jth point in v
                             // direction
    );
```

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

Description: Refer to Action.

Errors: Returns –1 on bad input: null surface or negative indices.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_ij_knv

Function: Spline Interface, Construction Geometry

Action: Gets the *i*th knot in *v* direction.

Prototype:
```
double bs3_surface_ij_knv (
    bs3_surface bs,          // given surface
    int i,                   // ith point in u
                             // direction
    int j                    // jth point in v
                             // direction
    );
```

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

Description: Refer to Action.

Errors: Returns –1 on bad input: null surface or negative indices.

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_init

Function: Spline Interface, Construction Geometry

Action: Initializes the spline surface system.

Prototype:
```
void bs3_surface_init ();
```

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

Description: ACIS calls this routine once; it should *not* be called more than once.

Errors: None

Limitations: None

Library: kernel

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

Effect: System routine

bs3_surface_interp_knots

Function: Spline Interface, Construction Geometry

Action: Creates a bicubic surface that interpolates or fits a set of points, with specified boundary derivatives and twist vectors.

Prototype:

```
bs3_surface bs3_surface_interp_knots (
    int nu,                 // number points in u
    int nv,                 // number points in v
    SPAposition* points,    // position array
    double knotsu[],        // u knot array
    double knotsv[],        // v knot array
    SPAvector* deru,        // u derivative array of
                            // end conditions
    SPAvector* derv,        // v derivative array of
                            // end conditions
    SPAvector deruv[]       // uv derivative array of
                            // corner twist vectors
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/vector.hxx"
#include "kernel/spline/bs3_srf/bs3surf.hxx"
#include "kernel/spline/d3_bs3/spd3rtn.hxx"
```

Description:

This function interpolates or fits an array of positions, using knot vectors in the u and v directions, boundary derivatives (i.e., tangent vectors) and twist vectors.

nu is the number of positions in the u direction, nv the number of positions in the *v* direction. points is a two dimensional array of positions [nu*nv]. knotsu is the *u* knot vector, knotsv the *v* knot vector. Initially there is a one–to–one correspondence between knots and positions; however, it is required that the –1, –2, *n* and *n*+1 element of the knot vectors be addressable, though they need not be set. Therefore the knotsu and knotsv should both be of size [nu+4]. The point for *u* knot *i* and *v* knot *j* is points [j*nu + i]. The increasing *u* values are stored contiguously.

deru contains the start and end derivatives for the *u* direction, elements 0 to nv–1 for the bottom *u* knot end, and elements nv to 2*nv–1 for the top end. Specifying a zero length vector means that natural end conditions will be used for that place on the surface. derv and deru work the same way, replacing nv by nu.

Finally, deruv is an array of four vectors giving the twist vectors, i.e., the *uv* cross derivative at the corners of the domain. They are stored in the following order:

```
(lo_u,lo_v), (hi_u,lo_v), (lo_u,hi_v), (hi_u,hi_v)
```

As with first derivatives along the boundaries, a zero length vector is an indicator to use natural end conditions.

The interpolation algorithm assumes that all the knot values are distinct, the surface is open in both directions, and that there is generally nothing unusual going on.

As a side effect of the interpolation algorithm, this routine will set any vectors in derv which you have passed in as zero vectors. deru is left alone.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/d3_bs3/spd3rtn.hxx

Effect: Changes model

bs3_surface_intp

Function: Spline Interface, Construction Geometry

Action: Interpolates a mesh of points.

Prototype:
```
bs3_surface bs3_surface_intp (
    int num_u,              // number of points in u
    int num_v,              // number of points in v
    const SPAposition pts[],// points [num_u][num_v]
    const SPAunit_vector    // u derivatives at start
        du_s[],             // [num_v]
    const SPAunit_vector    // u derivatives at end
        du_e[],             // [num_v]
    const SPAunit_vector    // v derivatives at start
        dv_s[],             // [num_u]
    const SPAunit_vector    // v derivatives at end
        dv_e[]              // [num_u]
    );
```

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

Description: Interpolates a mesh of points, optionally specifying the *u* and *v* tangent directions along the edges of the surface.

The points are stored in a single dimension array with *u* varying first, then *v*. The interpolation scheme is cubic in both the *u* and *v* directions. The start and end derivatives must all be specified or all be NULL.

Errors: None

Limitations: No two adjacent points to be interpolated can be same with in tolerance. However the interpolated points can be same at start and end for closed surfaces, so other types of configurations are allowed which result in self–intersecting surfaces.

Library: kernel

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

Effect: Changes model

bs3_surface_invdir

Function: Spline Interface, Construction Geometry

Action: Gets the direction in the parameter space of a surface at a given position that corresponds to a given object-space tangent direction.

Prototype:
```
SPApar_dir bs3_surface_invdir (
    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: Usually, this routine will normalize the result of calling bs3_surface_unitvec.

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

Limitations: Results are not defined for points that do not lie on the surface.

Library: kernel

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

Effect: Read–only

bs3_surface_invert

Function: Spline Interface, Construction Geometry

Action: Gets the parameter of a point on a 3D B-spline surface.

Prototype:
```
SPApar_pos bs3_surface_invert (
    SPAposition const& pos, // given point
    bs3_surface bs,         // given surface
    SPApar_pos const& uv    // uv guess
        =*(SPApar_pos*)NULL_REF
    );
```

Includes:
```
#include "kernel/acis.hxx"
#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: If initial parameter values are given, they may be assumed to be close to the desired point compared with any other point of (local) minimum distance, so there is no requirement to check that the value obtained is indeed the nearest point.

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

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_join_u

Function: Spline Interface, Construction Geometry

Action: Joins two compatible three–dimensional B–spline surfaces together.

Prototype:
```
bs3_surface bs3_surface_join_u (
    bs3_surface first_part, // left-hand surface
    bs3_surface last_part   // right-hand surface
    );
```

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

Description: Joins two compatible three–dimensional B–spline surfaces together (without checking for compatibility). The surfaces are joined along the high u–parameter edge of the first, and the low u–parameter edge of the second.

Errors: None

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_join_v

Function: Spline Interface, Construction Geometry

Action: Joins two compatible three–dimensional B–spline surfaces together.

Prototype:
```
bs3_surface bs3_surface_join_v (
    bs3_surface first_part, // left-hand surface
    bs3_surface last_part   // right-hand surface
    );
```

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

Description: Joins two compatible three–dimensional B–spline surfaces together (without checking for compatibility). The surfaces are joined along the high v–parameter edge of the first, and the low v parameter edge of the second.

Errors: None

Limitations: None

Library: kernel

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

Effect: Changes model

bs3_surface_knots_u

Function: Spline Interface, Construction Geometry

Action: Gets the number of knots in the *u* direction and the knot values in the *u* direction, for the given surface.

Prototype:
```
void bs3_surface_knots_u (
    bs3_surface bs,          // input surface
    int& num_knots_u,        // number of knots
    double*& uknots          // knot vector
    );
```

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 knot points in the *u* direction for the given surface. The knot multiplicity (i.e., the number of knots in the array with the same value), will be equal to the degree plus one at both ends of the array. It is the responsibility of the calling application to delete the knot array.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_knots_v

Function: Spline Interface, Construction Geometry

Action: Gets the number of knots in the *v* direction and the knot values in the *v* direction, for the given surface.

Prototype:
```
void bs3_surface_knots_v (
    bs3_surface bs,          // input surface
    int& num_knots_v,        // number of knots
    double*& vknots          // knot vector
    );
```

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 knot points in the *v* direction for the given surface. The knot multiplicity (i.e., the number of knots in the array with the same value), will be equal to the degree plus one at both ends of the array. It is the responsibility of the calling application to delete the knot array.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_knottol

Function: Spline Interface, Construction Geometry

Action: Gets the parametric criterion used to decide whether a given parameter is a knot.

Prototype:
```
double bs3_surface_knottol ();
```

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

Description: This routine is for the purpose of choosing between discontinuous "sided" derivatives.

Errors: None

Limitations: None

Library: kernel

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

Effect: Read–only

bs3_surface_loft_curves

Function: Spline Interface, Construction Geometry

Action: Lofts a surface from an array of n bs3_curves.

Prototype:
```
bs3_surface bs3_surface_loft_curves (
    bs3_curve*  curves[],    // curves
    double knots[],          // knots
    double fitol,            // fit tolerance
    int n,                   // position
    double& actual_tol       // returns the actual
                             // tolerance used
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_crv/bs3curve.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/d3_bs3/spd3rtn.hxx”
```

Description: The array contains n+2 curves, the nth and n+1th being derivative curves to give the correct start and end derivatives while splining across the control points of the given curves. The curves proper are stored in elements 0 to n–1 inclusive. This does some casting of positions to vectors which isn’t quite proper. It doesn’t worry about potential periodicity or any other such details. The given knot vector becomes the u knot vector of the surface.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/d3_bs3/spd3rtn.hxx

Effect: Changes model

bs3_surface_loft_u_curves

Function: Spline Interface, Construction Geometry

Action: Lofts a series of similar bs3_curves into a bs3_surface.

Prototype:
```
bs3_surface bs3_surface_loft_u_curves (
    int n_crvs,              // number of curves
    bs3_curve crvs[],        // array of curves
    double knots[]           // array of knots
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/spline/bs3_crv/bs3curve.hxx”
#include ”kernel/spline/bs3_srf/bs3surf.hxx”
#include ”kernel/spline/d3_bs3/spd3rtn.hxx”
```

Description: Loft a series of similar bs3_curves into a bs3_surface by splining across the control points of the curves with a cubic interpolation, and using the given knot vector which will become the *v* knot vector of the surface. The knot vector of the curves becomes the u knot vector of the surface. Closure forms both left as open.

This function is essentially the same as bs3_surface_loft_curves. However it does it with *u* and v transposed. So the supplied curves become the u parameter curves, and the longitudinal direction becomes the v direction. n_crvs is the number of curves not including the two derivative curves, which are the final two of the array. knots must be addressable 2 above and 2 below its start.

Errors: None

Limitations: None

Library: kernel

Filename: kern/kernel/spline/d3_bs3/spd3rtn.hxx

Effect: Changes model