Chapter 3.
Functions DM Aa thru Fz

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.

DM_add_area_cstrn

Function: Deformable Modeling, DML Constraints

Action: Applies an area constraint that fixes a subarea of a deformable model.

Prototype:

```
int DM_add_area_cstrn (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: member of target
                               // dmod hierarchy,
                               // pre–allocated
    int tag_flag,              // in : specify tgt dmod
                               // 1 = active dmod
                               // 2 = root dmod
                               // else = member dmod
                               // with tag==tag_flag
    int zone_flag,             // in: 0=zone area moves
                               // zone compliment fixed
                               // 1 =zone area fixed
                               // zone compliment moves
    DS_zone* zone,             // in: defines
                               // constraint partition,
                               // [nested]
    void* src_data             // opt: app data stored
        = NULL,                // with area_cstrn,
                               // [pass–thru]
    int tag                    // opt: when = –1 assign
        = –1,                  // next tag number else
                               // use this tag number
    SDM_options* sdmo          // in:SDM_options pointer
        = NULL                 // note: sets active dmod
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”dshusk/dskernel/dmapi.hxx”
#include ”dshusk/dskernel/dsdmod.hxx”
#include ”dshusk/dskernel/dszone.hxx”
#include ”dshusk/dskernel/sdm_options.hxx”
```

Description: Builds and adds to the target deformable model an area constraint and returns the integer tag value for that constraint. An area constraint implements an isolated deformation by partitioning the shape of the deformable model into two pieces, a piece that is fixed and a piece that can be deformed.

The partitioned area is represented by a DS_zone structure. A zone can be created by calling the function DM_build_square_zone(). The zone marks off a region of the surface’s domain.

The input pointer, dmod, is a pointer to any deformable model within the deformable modeling hierarchy. The input tag_flag, specifies which deformable model in the hierarchy is the target for the area constraint.

When tag_flag = 1, the target is the active deformable model. When tag_flag = 2, the target is the root sibling deformable model. Else the target is the deformable model whose tag id equals the tag_flag value.

The input zone_flag value specifies which of the two shape partitions defined by the input zone is fixed.

When zone_flag = 0, the zone exterior is fixed. When zone_flag = 1, the zone's interior is fixed. See the comments under Limitations on element locking.

The input src_data, is for the convenience of the application so that the application can store whatever additional information with the constraint that it desires. The deformable modeling package never accesses this pointer. If an application uses this pointer, the application has to free the memory associated with it prior to deleting the area constraint to prevent a memory leak.

When the tag value is –1, the deformable modeling library assigns a unique tag value to the newly constructed constraint, otherwise the input tag value is assigned to the newly constructed area constraint. Applications need to ensure that assigned tag values are unique to the entire deformable modeling hierarchy that contains the area constraint.

Errors: Sets rtn_err to 0 for success, else

DM_parse_tag_flag()
errors,

DM_NULL_INPUT_PTR
when model or zone are NULL on input,

DM_DMOD_MISSING_PFUNC
when model does not contain a valid DS_pfunc object.

DM_MIXED_AREA_CSTRN_DIM
when the domain_dim of zone != the model's domain_dim.

DM_BAD_ZONE_FLAG_VALUE
when zone_flag is not one of 0=zone area moves or 1=zone area fixed.

DM_BAD_TAG_VALUE
when the input tag number is not –1 or larger than 0.

Limitations: Only rectangular zones can be used to build an area constraint. With NUBS or NURBS, area constraints will lock, or fix, areas corresponding to entire elements; an element is a rectangle bounded by adjacent break–point (i.e., knot) parameter values. The actual area fixed corresponds to the minimal set of elements which contains the zone–fixed subdomain. A rule of thumb is that the zone subdomain fixes the elements it touches. When the zone_flag is 1, all elements touched by the open interior of the zone subdomain are fixed. When the zone_flag is 0, all elements touched by the open exterior of the zone subdomain are fixed. (Open means we exclude the boundary.) For example, consider an area constraint with its zone_flag 0, on a cubic NUBS or NURBS. In this case, the zone exterior is fixed, and the zone interior must contain at least a 4 by 4 array of elements to be able to deform. Typically we add zone constraints to cubic NUBS or NURBS which have more than 10 knots in each parameter.

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_attractor

Function: Deformable Modeling, DML Loads

Action: Adds an attractor to a deformable model and returns the new tag identifier or negative error code.

Prototype:

```
int DM_add_attractor (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: member of target
                            // dmod hierarchy to
                            // search
    int tag_flag,           // in: specify tgt dmod
                            // 1 = active dmod
                            // 2 = root dmod
                            // else = member dmod
                            // with tag=tag_flag
    double* image_pt        // in: attractor's
        = NULL,             // location or NULL
                            // sized:[image_dim]
                            // mem:[not–nested]
    int power               // input measure of
        = 2,                // load's locality
                            // 0,1=global: =2,3,...
                            // more local
    double gain             // in: magnitude of
        = 0.0,              // load gain
    int tag                 // in: when = –1 assign
        = –1,               // next tag number
                            // else use this
                            // tag number
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              // note: sets active dmod
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Modifies the target deformable model.

Builds and adds to the target deformable model an attractor load specified by an image point location. When image_pt is NULL the attractor location is computed as an offset in the surface normal direction from the center of the target deformable model.

An attractor acts like a concentrated charge either attracting (negative gains) or repulsing (positive gains) the entire deformable model shape with a 1/distance**2 law. The distance is measured from the attractor's location to each point on the deformable model.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag)

Returns a new load's tag number when successful.

Errors: DM_parse_tag_flag() errors

DM_BAD_TAG_VALUE
The tag value must be –1, to have the system assign one, or 2 or greater.

DM_NO_ROOT_DMOD
The root deformable model cannot be NULL on input.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_crv_cstrn

Function: Deformable Modeling, DML Constraints

Action: Adds a curve constraint to the target model and returns a new tag identifier or a negative error code.

Prototype:

```
int DM_add_crv_cstrn (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: member of target
```

```
                                // model hierarchy
int tag_flag,                   // in: specify tgt model
                                // 1 = active model
                                // 2 = root model
                                // else = member model
                                // with tag=tag_flag
DS_dmod* parent_dmod            // opt: Seam's parent
    = NULL,                     // shape for hierarchies.
                                // [not-nested]
int domain_flag                 // in : 0=src_C_pfunc in
    = 0,                        // orig_dmod_space
                                // 2=src_C_pfunc in
                                // internal_pfunc_space
DS_pfunc* src_C_pfunc           // opt: uv domain curve
    = NULL,                     // description.[nested]
                                // input in
                                // orig_dmod_space.
DS_pfunc* src_W_pfunc           // opt: cstrn pos shape,
    = NULL,                     // [nested]
DS_pfunc* src_Wn_pfunc          // opt: cross-tang shape,
    = NULL,                     // [nested].
DS_pfunc* src_Wnn_pfunc         // opt: curvature shape,
    = NULL,                     // [nested].
void(* src_CW_func)             // opt: user given
    (void* src_data,            // src_crv description
    double s,                   // in: curve s param
                                // value
double* C,                      // out: surface C=[u,v]
                                // point value
double* Cs,                     // out: surface dC/ds vec
                                // in surf for s
double* W,                      // out: image space
                                // W=[x,y,z] point val
double* Wu,                     // out: image space dW/du
double* Wv,                     // out: image space dW/dv
double* Wuu,                    // out: image space
                                // d2W/du2
double* Wuv,                    // out: image space
                                // d2W/dudv
double* Wvv,                    // out: image space
                                // d2W/dv2
double& dist2)                  // out: dist**2
    = NULL,                     // (xyz,proj_to_3d(uv))
void* src_data                  // src_CW_func
    = NULL,                     // callback data
```

```
    DS_CSTRN_SRC src_type    // in: records cstrn
        =  ds_user_cstrn,    // one of:
                             // ds_solid_cstrn=
                             // shared topo bndry
                             // ds_bound_cstrn=
                             // unshared topo bndry
                             // ds_user_cstrn =
                             // user created cstrn
                             // ds_seam_cstrn =join 2
                             // hierarchical faces
    int behavior             // OR of:
        = (1 << 3),          // DM_POS_FIXED
                             // DM_TAN_FIXED
                             // DM_CURV_FIXED
    double tang_gain         // UNUSED
        = 1.0,               // UNUSED
    int tag                  // in: when == –1 assign
        = –1,                // next tag number else
                             // use this tag number
    SDM_options* sdmo        // in:SDM_options pointer
        = NULL               // note: sets active dmod
    );
```

Includes:

```
#include ″kernel/acis.hxx″
#include ″dshusk/dskernel/dmapi.hxx″
#include ″dshusk/dskernel/dmapinum.hxx″
#include ″dshusk/dskernel/dsdmod.hxx″
#include ″dshusk/dskernel/dspfunc.hxx″
#include ″dshusk/dskernel/sdm_options.hxx″
```

Description: Modifies the target deformable model.

Builds and adds a curve constraint to the target deformable model. The shape of the curve constraint is really two curves, one in the original domain space of the target deformable model and another curve shape specified within the target deformable model’s image space. The shape of the curve in the target deformable model’s original domain space is given by C(s) = [u(s), v(s)] where each valid curve parameter value, s, is used to compute curve positions u and v. The curve positions u and v must be returned in the original domain space of the constrained deformable model. The shape of the curve in three dimensional image space is given by W(s) = [x(s), y(s), z(s)] where each valid curve parameter value, s, is used to compute curve positions x, y, and z.

The parametric shape of the curve can be given in two ways: either by an explicit curve represented by a DS_pfunc (the src_C_pfunc input argument), or by a callback function (the src_CW_func input argument) but not by both. That is, only one of src_C_pfunc and src_CW_func can be something other than NULL on input. The src_CW_func must specify the shape of the constraint curve in the orig_dmod_space. That is each call to src_CW_func(s) must return C and Cs values as measured in the domain space of the deformable model at the time it was constructed. When the domain curve shape is specified by src_C_pfunc, the domain_flag value is used to determine in which space the curve was built. When domain_flag == 0, the src_C_pfunc's image_space is equal to the orig_dmod_space. When domain_flag == 2, the src_C_pfunc's image_space is equal to the internal_pfunc_space. When src_C_pfunc is supplied in the orig_dmod_space, it is scaled by this function from the orig_dmod_space to the internal_pfunc_space and its pointer is stored internally. src_C_pfunc may not be described to this function in unit_space.

The image space shape of the curve can be defined in three ways defining three different types of curve constraints. Seam constraints (ds_seam_cstrn) are used to connect the constrained deformable model patch to its parent. User constraints (ds_user_cstrn) are used to allow the user to specify any curve within the deformable model to be constrained from further motion. And three dimensional curve constraints in which the curve shape is defined by an external curve, typically a curve in a solid model. There are two types of three dimensional curves: ds_bound_cstrn in which the boundary is not shared by any other faces as in the boundary of a sheet model, and ds_solid_cstrn in which the boundary is shared by other faces as with 2 manifold models.

The image space shape of a seam constraint is computed by evaluating the shape of the parent deformable model. For seam constraints the parent_dmod argument must be something other than NULL. The curve constraint for a seam must be built in the parent's original domain range.

The image space shape of a user constraint is computed by evaluating the shape of the deformable model or by evaluating an explicit function given by the src_W_pfunc argument. The src_W_pfunc argument is optional for user constraint curves. Similarly, the cross–tangent and the cross–curvature values can be specific functions represented by DS_pfunc objects in src_Wn_pfunc and src_Wnn_pfunc. When supplied and the behavior of the curve constraint is set to DM_POSITION, DM_TANG and/or DM_CURVATURE the shape of the constrained shape is deformed to satisfy the constraints,

W(C(s)) = src_W_pfunc(s) (an image_space vector function)
Wn(C(s)) = src_Wn_pfunc(s) and (an image_space vector function)
k(C(s)) = src_Wnn_pfunc(s) (a scalar function)

Where s is the parameter variable used to describe point locations on the const

Wn(s) is the cross tangent vector for value s
Wnn(s) is the cross 2nd parametric derivative for value s
Wu(s) is the u direction parametric derivative dW/du for value s
Wv(s) is the v direction parametric derivative dW/dv for value s
k(s) = size(cross(Wn,Wnn)/ size (Wn)**3 = the cross curvature value for value and
n(s) = normalized (cross(Wu,Wv)) = the surface normal for value s

The image space shape of a three dimensional curve constraint (ds_solid_cstrn or ds_bound_cstrn) is always defined by the src_CW_pfunc callback in which case the parent_dmod, src_C_pfunc, and src_W_pfunc are all set to NULL. The calling program is required to supply the callback function. For ACIS applications the callback is already written and is called, DS_true_edge_eval and the src_data pointer is populated by the class DS_2acis. The callback function takes as input a curve parameter value, s, and returns both a domain point and an image point value. The domain point value has to be returned in the constrained deformable model's original domain space.

The constraint may act to constrain any combination of the shape's position, tangent, and curvature along the length of the curve constraint determined by the behavior bit array value. The behavior bits are named DM_POS_FIXED, DM_TAN_FIXED, and DM_CURVATURE. The surface tangent and curvature constraints act in the direction perpendicular to the curve. Combining DM_POSITION and DM_TANG behaviors allows seam constraints to be C1 continuous with their parents. Combining DM_POS_FIXED, DM_TAN_FIXED, and DM_CURVATURE behaviors allows seam constraints to be C1 continuous with their parents. Combining DM_POSITION, DM_TANG, and DM_CURVATURE behaviors allows seam constraints to be C2 continuous with their parents. To be C1 continuous tang_gain must be 1.0.

When the tag value is –1 the system selects a unique tag number to give to the newly created curve constraint. Otherwise, it uses the given tag value, but never checks that tag value for uniqueness.

Inherited behaviors: Constraints carry with them a set of behavior states including position, tangent, deletable, and stoppable. The position and tangent states are set by the behavior input argument. A deletable constraint may be removed by an end user at run time. A stoppable constraint may be disabled by an end user at run time. The deletable and stoppable behaviors are given based on the src_type as shown below. Also included in the table is a summary of which kinds of constraints will use the callback function and the source data.

DS_CSTRN_SRC	deletable	stopable	Src_data	Src_CW_func
ds_solid_cstrn	NO	NO	YES	YES
ds_bound_cstrn	NO	YES	YES	NO
ds_user_cstrn	YES	YES	NULL	NO
ds_seam_cstrn	NO	NO	YES	NO

ds_solid_cstrn = curve is from a solid model boundary and any number of faces may connect to the curve.

ds_bound_cstrn = curve is from a sheet model boundary and only the face being deformed connects to crv.

ds_user_cstrn = curve is from a user interface and not intended to represent a topology boundary from a B–rep model.

ds_seam_cstrn = curve is used to join 2 hierarchical faces in a deformable modeling hierarchy.

Returns a tag identifier for newly created curve constraint.

The basic flavor for using a constraint is:

1. Create and apply the constraint to a deformable model,
   (DM_add_pt_cstrn()),
   (DM_add_crv_cstrn()),
   (DM_add_link_cstrn()),
   (DM_add_area_cstrn()).
2. Select which geometry properties are being constrained,
   (DM_set_cstrn_behavior()).
3. For pt_cstrns, modify the geometry property values by moving the point constraint display points as described above,
   (DM_set_pt_xyz()).
4. Call solve to see how the constraint modifications changed the deformable model shape,
   (DM_solve()).
5. Optionally, for pt_cstrns, the point constraint's domain position and domain directions (for surfaces) may be modified with
   (DM_set_pt_uv()),
   (DM_set_cstrn_pttan_uv_dir()).
6. Optionally, modify the constraint rendering and image_pt locations to optimize viewing and/or interactions,
   DM_set_tan_display_gain()
   DM_set_comb_graphics().

Errors: DM_parse_tag_flag() errors

DM_NO_ROOT_DMOD – The root deformable model cannot be NULL on input.

DM_BAD_SRC_CURVE_TYPE – The src_type must be one of ds_solid_cstrn, ds_bound_cstrn, ds_user_cstrn, or ds_seam_cstrn.

DS_MISSING_SRC_DATA – The curve's source type conflicts with the particular set of input values given.

DM_MIXED_CRV_CSTRN_DIM – The image_dim of src_C_pfunc must equal the deformable model's domain_dim.

DM_BAD_CRV_CSTRN_DIM – The input src_C_pfunc must be a curve, i.e. its domain_dim must be 1.

DM_BAD_DOMAIN_PT_RANGE – The domain point must be completely contained by the deformable model.

DM_BAD_SRC_PFUNC_MAPPING – Whenever a src_pfunc does not have a proper domain_dim or image_dim value. The domain_dim value for all src_pfuncs must be 1 (they are all curves), and the image_dim for the src_W and src_Wn DS_pfuncs must equal the image_dim of the object being constrained, and the image_dim of the src_Wnn DS_pfunc must be 1.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_crv_load

Function: Deformable Modeling, DML Loads

Action: Adds a distributed curve spring and returns a new tag identifier or negative error code.

Prototype:

```
int DM_add_crv_load (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: member of tgt dmod
                               // hierarchy to search
    int tag_flag,              // in: specify tgt dmod
                               // 1 = active dmod
                               // 2 = root dmod
                               // else = member dmod
                               // with tag=tag_flag
    int domain_flag,           // in: 0=src_C_pfunc in
                               // orig_dmod_space
                               // 2=src_C_pfunc in
                               // internal_pfunc_space
    DS_pfunc* src_C_pfunc,     // in: domain space curve
                               // connected to image
                               // curve over common
                               // s values. input in
                               // orig_dmod_space.
    DS_pfunc* src_W_pfunc      // in: the image
        = NULL,                // space curve
                               // if NULL project
                               // src_C_pfunc (both
                               // curves domain_dim
                               // = 1)
    double gain                // n: stiffness
        = 0.0,                 // connecting 2 curves
    int tag                    // in: when = -1 assign
        = -1,                  // next tag number
                               // else use this
                               // tag number
    SDM_options* sdmo          // in:SDM_options pointer
        = NULL                 // note: sets active dmod
    );
```

```
int DM_add_crv_load (
    int& rtn_err,                 // rtn: new tag id or
                                  // negative err code
    DS_dmod* dmod,                // member of target
                                  // dmod hierarchy
    int tag_flag,                 // specify tgt dmod
                                  // 1 = active dmod
                                  // 2 = root dmod
                                  // else = member dmod
                                  // with tag==tag_flag
    DS_dmod* parent_dmod          // shape's parent
        = NULL,                   // shape for seams
    int domain_flag               // 0=src_C_pfunc in
        = 0,                      // orig_dmod_space
                                  // 2=src_C_pfunc in
                                  // int pfunc space
    DS_pfunc* src_C_pfunc         // uv domain curve
        = NULL,                   // description input
                                  // in orig_dmod_space
    DS_pfunc* src_W_pfunc         // xyz image curve
        = NULL,                   // description
    DS_pfunc* src_Wn_pfunc        // xyz image curve
        = NULL,                   // description
    DS_pfunc* src_Wnn_pfunc       // xyz image curve
        = NULL,                   // description
    void(* src_CW_func)           // user given src_crv
        (void* src_data,          // description
        double s,                 // in : curve s param
                                  // value
        double* C,                // out: surface
                                  // C=[u,v] pt. value
        double* Cs,               // out: surface dC/ds
                                  // vec in surf for s
        double* W,                // out: image space
                                  // W=[x,y,z] pt. val
        double* Wu,               // out: image space
                                  // dW/du for s
        double* Wv,               // out: image space
                                  // dW/dv for s
        double* Wuu,              // out: image space
                                  // d2W/du2
        double* Wuv,              // out: image space
                                  // d2W/dudv
        double* Wvv,              // out: image space
                                  // d2W/dv2
```

```
    double& dist2)          // out: dist**2
    = NULL,                 // (xyz,proj_to_3d
                            // (uv))
void* src_data              // opt: passed to
    = NULL,                 // src_CW_func
                            // callback
DS_CSTRN_SRC src_type       // in : records cstrn
    = ds_user_cstrn,        // origin
int behavior                // OR of:
     =(1 << 3),             // DM_POS_FIXED
                            // = surf pos cnstrn
                            // DM_TAN_FIXED =
                            // surf tan cnstrn
                            // DM_CURV_FIXED =
                            // surf curv cnstrn
double gain                 // in : stiffness
    = 0.0,                  // connecting the 2
                            // curves
int tag                     // in : when == –1
    = –1,                   // assign next tag
                            // number, else use
                            // this tag number
SDM_options* sdmo           // in:SDM_options
    = NULL                  // pointer
                            // note: sets active
                            // dmod
);
```

Includes:

```
#include ″kernel/acis.hxx″
#include ″dshusk/dskernel/dmapi.hxx″
#include ″dshusk/dskernel/dsdmod.hxx″
#include ″dshusk/dskernel/dspfunc.hxx″
#include “dshusk/dskernel/dmapinum.hxx″
#include ″dshusk/dskernel/sdm_options.hxx″
```

Description: Modifies the target deformable model.

DS_CSTRN_SRC src_type must be one of the following values:

ds_solid_cstrn = shared topo boundary
ds_bound_cstrn = unshared topo boundary
ds_user_cstrn = user created constraint
ds_seam_cstrn = join two hierarchical faces

Builds and adds a curve load to the target deformable model using src_C_pfunc, free_crv, and the gain input arguments. A curve load is a distributed spring which connects a set of points on the deformable model to a set of points in three dimensional space. The deformable model points are specified by a domain space curve, C(s) = [u(s), v(s)] where s is the curve's parameter. The domain space curve may be specified in either the loaded deformable model's original domain space (orig_dmod_space) or in the internal pfunc's domain space (internal_pfunc_space). Set domain_flag = 0, when src_C_pfunc is specified in orig_dmod_space and set domain_flag = 1, when src_C_pfunc is specified in internal_pfunc_space. When domain_flag == 0, the src_C_pfunc is scaled from the orig_dmod_space to the internal_pfunc_space and its pointer is stored internally. The src_C_Pfunc may not be described to this function in a unit_space domain range.

The free space points are specified by an image space curve, W(s) = [x(s), y(s), z(s)] where s is the same parameter used for the domain space curve. For every curve parameter value, s, there is a point–to–point spring between the domain curve point on the surface and the image curve point in free space. The net effect is a curve spring load or an elastic membrane stretched between the two curves. The two curves, Src_C_pfunc and free_crv should share a common domain range. When src_W_pfunc, is NULL on entry it will be generated by projecting the src_C_pfunc through the deformable model's shape into image space.

The image space of the Src_C_pfunc is the same as the domain_space of the loaded pfunc. This function does NOT scale the Src_C_pfunc from the unit square domain into the pfunc's domain.

This function has two signatures. The first signature (with only eight arguments) is supplied for backward compatibility. The second signature allows the freedom in specifying target shape and properties to be loaded; the user can load position, tangent, or curvature. It is intentionally modeled on the signature of DM_add_crv_cstrn; see that routine's documentation for a discussion the argument list.

Even though multiple behaviors can be associated with one curve load, it is strongly recommended that a different curve load be added for each behavior to be controlled. Each behavior will then have a separate tag which can be used to separately tune how closely the curve is pulled to position tangency targets.

gain is the stiffness of the elastic membrane.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag)

Returns a the tag number of the newly created load.

Errors: DM_parse_tag_flag()
errors:

DM_BAD_TAG_VALUE
The input tag value must be –1 or positive.

DM_NO_ROOT_DMOD
The the input deformable model cannot be NULL.

DM_NULL_INPUT_PTR
The input src_C_pfunc is NULL on entry.

DM_MIXED_CRV_LOAD_DIM
The src_C_pfunc's image_dim must be equal to deformable model's domain_dim.

DM_MIXED_FREE_LOAD_DIM
The src_W_pfunc's image_dim must be equal to the deformable model's image_dim.

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_curve_patch

Function: Deformable Modeling, DML Patches

Action: Makes and adds a patch- to-patch hierarchy and returns a new patch tag identifier or an error.

Prototype:

```
int DM_add_curve_patch (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: dmod to be the
                            //parent
    int domain_flag,        // in : 0=domain_pts in
                            // orig_dmod_space,
                            // 1=domain_pts in
                            // unit_space.
                            // 2=domain_pts in
                            // internal_pfunc_space.
    double min,             // in: min–domain
                            // (0 to 1 range)
    double max,             // in: max–domain
                            // (0 to 1 range)
    int refinement,         // in:
                            // parent_knot_spacing/
                            // child_knot_spacing
    void* patch_entity,     // in: app entity ptr
                            // stored with patch
    void* seam_data[2],     // in: app entity data
                            // stored with each seam
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”dshusk/dskernel/dmapi.hxx”
#include ”dshusk/dskernel/dsdmod.hxx”
#include ”dshusk/dskernel/sdm_options.hxx”
```

Description: Modifies the target deformable model.

Adds a patch to the input deformable model. The domain of the patch is a span connected to its parent by a seam which consists of two point constraints. The patch may be made to connect to its parent with C0 (position only) or C1 (position and tangent) continuity but is returned by this function with C1 continuity.

The size of the span is specified by the minimum and maximum u values which are given in the parent's domain space. When domain_flag is 1 the min and max u values are specified in a range that varies from 0 to 1 (and is scaled to the parent's dmod domain space in this function), and when domain_flag is 0 the min and max u values are specified in the parent's dmod domain space. When domain_flag is 2, the min and max u values are specified in the parent's pfunc's domain space.

The refinement argument specifies the number of control points in the child patch compared to the parent patch. For example, a refinement value of 2 will double the density of control points in the child compared to the parent. Refinement must be an integer value greater than 0. The patch entity pointer is made available to applications. It is stored with the patch's deformable model data structure but is never accessed. Applications can retrieve this pointer with the deformable model method call, DS_deformable model. Similarly, the seam data pointers are made available to applications to store data with each seam. These pointers are never accessed but can be accessed by applications with the call, DS_deformable model–>Seam(ii)–>Src_data();

Returns the tag of the newly created patch if successful, or an error.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_scale_unit_dpt_to_pfunc() errors

DM_UNCONTAINED_CHILD
The input patch boundary must be in the range of 0 to 1.

DM_BAD_REFINEMENT_VALUE
The input refinement must be larger than 0.

DM_PATCH_OCCLUDES_ROOT
The input patch completely covers the deformable model.

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_dist_press

Function: Deformable Modeling, DML Loads

Action: Adds a distributed pressure attribute to the deformable model and returns a new tag identifier.

Prototype:

```
int DM_add_dist_press (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: member of tgt dmod
                            // hierarchy to search
    int tag_flag            // in: specify tgt dmod
        = 1,                // 1 = active dmod
                            // 2 = root dmod
                            // else = member dmod
                            // with tag=tag_flag
    int domain_flag         // in: 0=domain_pts in
        = 1,                // orig_dmod_space
                            // 1=domain_pts in
                            // unit_space.
                            // 2=domain_pts in
                            // internal_pfunc_space.
    double* domain_min      // in: min-domain
        = NULL,             // [not-nested]
    double* domain_max      // in: max-domain
        = NULL,             // [not-nested]
    double gain             // in: magnitude of
        = 0.0,              // load gain
    int negate_flag         // in: change normal dir
        = 0,                // (1=negate,0=do not)
    int tag                 // in: when == -1 assign
        = -1,               // next tag number
                            // else use this
                            // tag number
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              // note: sets active dmod
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Modifies the target deformable model.

Builds and adds to the target deformable model a distributed pressure attribute with a gain specified by the input arguments. The distributed pressure applies a uniform pressure over a sub–rectangle of the deformable model's domain. That square is specified by its minimum and maximum corners given in the domain_min and domain_max. When domain_min and domain_max pointers are NULL, the distributed pressure is applied to the entire deformable model domain. When domain_flag is 1, domain_min and domain_max are specified in the unit domain range (values from 0 to 1) and when domain_flag is 0, domain_min and domain_max are given in the target dmod's original domain space (the range of knot values with which the target dmod was originally created). When domain_flag is 2, domain_min and domain_max are given in the target dmod's pfunc's internal domain space (the range of knot values used within the deformable modeler's internal calculations).

When tag equals –1 the next available tag number is assigned to the newly created load, otherwise the input tag number is used.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag)

Returns a newly created load's tag number.

Errors:

DM_parse_tag_flag()
errors

DM_scale_unit_dpt_to_pfunc()
errors

DM_BAD_TAG_VALUE
The input tag value must be –1 or positive.

DM_NO_ROOT_DMOD
The the input deformable model cannot be NULL.

DM_BAD_NEGATE_FLAG_VALUE
The negate_flag must be a 0 or a 1.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_link_C0_load

Function: Deformable Modeling

Action: Adds a link constraint to connect two root siblings to make two deformable models act as one.

Prototype:

```
int DM_add_link_C0_load (
    int& rtn_err,             // out: 0=success or neg
                              // err code
    int& tag_shift,           // out: dmod2's
                              // after_tags =
                              // before_tags +
                              // tag_shift
    DS_dmod* dmod1,           // in: specify 1st tgt
                              // dmod, [pre-alloced]
    DS_dmod* dmod2,           // in: specify 2nd tgt
                              // dmod, [pre-alloced]
    int domain_flag,          // in: 0=src_C_pfunc in
                              // orig_dmod_space
                              // 2=src_C_pfunc in
                              // internal_pfunc_space
    DS_pfunc* src1_C_pfunc,   // in: 0=src_C_pfunc in
                              // orig_dmod_space
                              // 2=src_C_pfunc in
                              // internal_pfunc_space
    DS_pfunc* src2_C_pfunc,   // in : uv domain curve2
                              // description, [nested]
                              // input in
                              // orig_dmod_space
    DS_pfunc* src1_W_pfunc    // opt: dmod1 cstrn pos
        = NULL,               // shape, [nested]
    DS_pfunc* src1_Wn_pfunc   // opt: dmod1 cross-tang
        = NULL,               // shape, [nested]
    DS_pfunc* src1_Wnn_pfunc// opt: dmod1 curvature
        = NULL,               // shape, [nested]
    DS_pfunc* src2_W_pfunc    // opt: dmod2 curvature
        = NULL,               // shape, [nested]
    DS_pfunc* src2_Wn_pfunc   // opt: dmod2 cross-tang
```

```
    = NULL,              // shape, [nested]
DS_pfunc* src2_Wnn_pfunc // opt: dmod2 curvature
    = NULL,              // shape, [nested]
void* src1_data          // opt: application data
    = NULL,              // stored for tgt1_dmod,
                         // [pass–thru]
void* src2_data          // opt: application data
    = NULL,              // stored for tgt2_dmod,
                         // [pass–thru]
void(* src_CW_func)      // opt: user given
    (void* src_data,     // src_crv description,
                         // [pre–alloced], with
                         // ACIS use:
                         // [DS_true_edge_eval]
                         // in: src_data for eval,
                         // [pre–alloced]
    double s,            // in : curve s param
                         // value, [pre–alloced]
    double* C,           // out: surface C=[u,v]
                         // point value,
                         // [pre–alloced]
    double* Cs,          // out: surface dC/ds vec
                         // in surf for s,
                         // [pre–alloced]
    double* W,           // out: image space
                         // W=[x,y,z] point val,
                         // [pre–alloced]
    double* Wu,          // out: image space dW/du
                         // for s, [pre–alloced]
    double* Wv,          // out: image space dW/dv
                         // for s, [pre–alloced]
    double* Wuu,         // out: image space
                         // d2W/du2
    double* Wuv,         // out: image space
                         // d2W/dudv
    double* Wvv,         // out: image space
                         // d2W/dv2
    double& dist2)       // out: dist**2
    = NULL,              // (xyz,proj_to_3d(uv))
double gain              // in : gain for load
    = 10000000.,         // behavior
int tag                  // in : when == –1 assign
    = –1,                // next tag number
                         // else use this tag
                         // number. note: sets
```

```
                                    // active dmod
    DM_flipped_state                // in: indicates
        flipped_state               // relative handedness of
        = DM_flip_unknown,          // two surfaces being
                                    // linked. sets active
                                    // dmod
    SDM_options* sdmo               // in:SDM_options pointer
        = NULL                      //
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/dmapinum.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Builds and adds to the dmod1 hierarchy a link that will cause dmod1 and dmod2 to deform simultaneously, and pulls them together with a spring load towards C0 continuity across this link. The input dmod1 and dmod2 must both be at the root levels of their deformable modeling hierarchies.

The difference between using this function and DM_add_link_cstrn() is in how the link constraint is enforced. This function uses a spring load (which is equivalent to a penalty method) for enforcing the constraint, while the DM_add_link_cstrn() call enforces the link link constraint with lag range variables. The difference is that the penalty method will tend to be more tolerant, which can be an advantage if the constraint is being enforced too stiffly to bend properly with the lag range constraint approach.

See the documentation for DM_add_link_cstrn() for a description of all the input argument uses.

Returns the tag id for the newly created link constraint

The handedness can be:

DM_unflipped	same handedness
DM_flipped	opposite handedness
DM_flip_unknown	unspecified by user, package will guess

Errors: Sets rtn_err to 0 for success, else

DM_parse_tag_flag()
errors,

DM_NULL_INPUT_PTR
when dmod1, dmod2 are NULL on input,

DM_NULL_INPUT_PTR
when src1_C_pfunc, or src2_C_pfunc == NULL on input and src_WC_func is NULL,

DM_MIXED_CRV_CSTRN_DIM
when the image_dim of src1_C_pfunc != the dmod1's domain_dim or src2_C_pfunc != the dmod2's domain_dim,

DM_BAD_CRV_CSTRN_DIM
when either the src1_C_pfunc or src2_C_pfunc are not a curve, i.e. one of their domain_dim's != 1.

DM_BAD_DOMAIN_PT_RANGE
when src1_C_pfunc is not contained by the tag1 dmod or src2_C_pfunc is not contained by the tag2 dmod.

DM_BAD_TAG_VALUE
when the input tag number is not –1 or larger than 0.

DM_NONMONOTONIC_LINK
when the link dcrvs, C1(s) or C2(s) are not described monotonically, that is dC/ds changes sign for some valid values of s.

DM_UNEVEN_LINK_SPEEDS
when the link_cstrn's dcrvs, C1(s0 and C2(s), represented by either src1_C_pfunc and src2_C_pfunc or by the callback src_CW_func, do not have a constantly related speed. That is if the requirement that dC1/ds = constant * dC2/ds is not true for all values of s.

DM_BAD_SRC_PFUNC_MAPPING
whenever a src_pfunc does not have a proper domain_dim or image_dim value. The domain_dim value for all src_pfuncs must be 1 (they are all curves), and the image_dim for the src_W and src_Wn DS_pfuncs must equal the image_dim of the object being constrained, and the image_dim of the src_Wnn DS_pfunc must be 1.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_link_C1_load

Function: Deformable Modeling, DML Constraints

Action: Adds a link constraint to connect two root siblings to make two deformable models act as one.

Prototype:

```
int DM_add_link_C1_load (
    int& rtn_err,             // rtn: new tag id or
                              // negative err code
    int& tag_shift,           // out: dmod2's
                              // after_tags =
                              // before_tags +
                              // tag_shift
    DS_dmod* dmod1,           // specify 1st tgt dmod
    DS_dmod* dmod2,           // specify 2nd tgt dmod
    int domain_flag,          // in : 0=src_C_pfuncs
                              // in orig_dmod_space
                              // 2=src_C_pfuncs in
                              // internal_pfunc_space
    DS_pfunc* src1_C_pfunc,   // uv domain curve
                              // description [nested]
                              // input in
                              // orig_dmod_space
    DS_pfunc* src2_C_pfunc,   // uv domain curve
                              // description [nested]
                              // input in
                              // orig_dmod_space
    DS_pfunc* src1_W_pfunc    // opt: cstrn pos shape
        = NULL,               // for dmod1, [nested].
    DS_pfunc* src1_Wn_pfunc   // opt: cross-tang shape
        = NULL,               // for dmod1, [nested].
    DS_pfunc* src1_Wnn_pfunc  // opt: curvature shape
        = NULL,               // for dmod1, [nested].
    DS_pfunc* src2_W_pfunc    // opt: cstrn pos shape
        = NULL,               // for dmod2, [nested].
    DS_pfunc* src2_Wn_pfunc   // opt: cross-tang shape
        = NULL,               // for dmod2, [nested].
    DS_pfunc* src2_Wnn_pfunc  // opt: curvature shape
        = NULL,               // for dmod2, [nested].
    void* src1_data           // application data
```

```
        = NULL,              // stored for tgt1_dmod
    void* src2_data          // application data
        = NULL,              // stored for tgt2_dmod
    void(* src_CW_func)      // function to
        (void* src_data,     // update
        double s,            // in: curve s param
                             // value
        double* C,           // out: surface C=[u,v]
                             // point value
        double* Cs,          // out: surface dC/ds vec
                             // in surf for s
        double* W,           // out: image space
                             // W=[x,y,z] point val
        double* Wu,          // out: image space dW/du
        double* Wv,          // out: image space dW/dv
        double* Wuu,         // out: image space
                             // d2W/du2
        double* Wuv,         // out: image space
                             // d2W/dudv
        double* Wvv,         // out: image space
                             // d2W/dv2
        double& dist2)       // out: dist**2
        = NULL,              // (xyz,proj_to_3d(uv))
    double gain              // strength of
        = 10000000.,         // load
    int tag                  // when = –1 assign next
        = –1,                // tag number
                             // else use this tag
                             // number
    DM_flipped_state         // in: indicates
        flipped_state        // relative handedness of
        = DM_flip_unknown,   // two surfaces being
                             // linked. sets active
                             // dmod
    SDM_options* sdmo        // in:SDM_options pointer
        = NULL               //
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”dshusk/dskernel/dmapi.hxx”
#include ”dshusk/dskernel/dsdmod.hxx”
#include ”dshusk/dskernel/dspfunc.hxx”
#include ”dshusk/dskernel/dmapinum.hxx”
#include ”dshusk/dskernel/sdm_options.hxx”
```

Description: Builds and adds to the dmod1 hierarchy a link that will cause dmod1 and dmod2 to deform simultaneously, and pulls them together with a spring load towards rotated C1 continuity across this link. The input dmod1 and dmod2 must both be at the root levels of their deformable modeling hierarchies. C1 as used here means tangent continuity only; position (C0) continuity is unaffected by this load.

Rotated C1 continuity is a generalization of C1 continuity. It is designed to account for differences in the parameterizations of the two surfaces. To enforce rotated C1 continuity, the surfaces are independently examined at every point along the link to determine the "cross–tangent derivative operator" at that point. The cross–tangent derivative operator is defined as the directional derivative (in *uv* space) for which the surface tangent vector is of unit length and perpendicular (inward) to the edge. This cross–tangent derivative operator is defined when the C1 behavior is turned on, and depends on the surface shape at the time it is defined. Locally rotated C1 is satisfied when the cross–tangent derivatives of the two surfaces are equal and opposite.

The difference between using this function and DM_add_link_cstrn() is in how the link constraint is enforced. This function uses a spring load (which is equivalent to a penalty method) for enforcing the constraint, while the DM_add_link_cstrn() call enforces the link link constraint with lag range variables. The difference is that the penalty method will tend to be more tolerant, which can be an advantage if the constraint is being enforced too stiffly to bend properly with the lag range constraint approach. In addition, link constraints are unable to enforce rotated C1 continuity; they enforce strict C1 continuity instead.

Refer to the documentation for DM_add_link_cstrn() for a description of all the input argument uses.

Returns the tag id for the newly created link constraint.

The handedness can be:

DM_unflipped	same handedness
DM_flipped	opposite handedness
DM_flip_unknown	unspecified by user, package will guess

Errors: Sets rtn_err to 0 for success, else

DM_parse_tag_flag()
errors,

DM_NULL_INPUT_PTR
When dmod1, dmod2 are NULL on input,

DM_NULL_INPUT_PTR
When src1_C_pfunc, or src2_C_pfunc == NULL on input and src_WC_func is NULL,

DM_MIXED_CRV_CSTRN_DIM
When the image_dim of src1_C_pfunc != the dmod1's domain_dim or src2_C_pfunc != the dmod2's domain_dim,

DM_BAD_CRV_CSTRN_DIM
When either the src1_C_pfunc or src2_C_pfunc are not a curve, i.e. one of their domain_dim's != 1.

DM_BAD_DOMAIN_PT_RANGE
When src1_C_pfunc is not contained by the tag1 dmod or src2_C_pfunc is not contained by the tag2 dmod.

DM_BAD_TAG_VALUE
When the input tag number is not –1 or larger than 0.

DM_NONMONOTONIC_LINK
When the link dcrvs, C1(s) or C2(s) are not described monotonically, that is dC/ds changes sign for some valid values of s.

DM_BAD_SRC_PFUNC_MAPPING
Whenever a src_pfunc does not have a proper domain_dim or image_dim value. The domain_dim value for all src_pfuncs must be 1 (they are all curves), and the image_dim for the src_W and src_Wn DS_pfuncs must equal the image_dim of the object being constrained, and the image_dim of the src_Wnn DS_pfunc must be 1.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_link_cstrn

Function: Deformable Modeling, DML Constraints

Action: Adds a link constraint to connect two root siblings to make two deformable models act as one.

Prototype:

```
int DM_add_link_cstrn (
    int& rtn_err,             // rtn: new tag id or
                              // negative err code
    int& tag_shift,           // out: dmod2's
                              // after_tags =
                              // before_tags +
                              // tag_shift
    DS_dmod* dmod1,           // specify 1st tgt dmod
    DS_dmod* dmod2,           // specify 2nd tgt dmod
    int domain_flag,          // in : 0=src_C_pfuncs
                              // in orig_dmod_space
                              // 2=src_C_pfuncs in
                              // internal_pfunc_space
    DS_pfunc* src1_C_pfunc,   // uv domain curve
                              // description [nested]
                              // input in
                              // orig_dmod_space
    DS_pfunc* src2_C_pfunc,   // xyz image curve
                              // description [nested]
                              // input in
                              // orig_dmod_space
    DS_pfunc* src1_W_pfunc    // opt: dmod1 cstrn pos
        = NULL,               // shape, [nested]
    DS_pfunc* src1_Wn_pfunc   // opt: dmod1 cross-tang
        = NULL,               // shape, [nested]
    DS_pfunc* src1_Wnn_pfunc  // opt: dmod1 curvature
        = NULL,               // shape, [nested]
    DS_pfunc* src2_W_pfunc    // opt: dmod2 cstrn pos
        = NULL,               // shape, [nested]
    DS_pfunc* src2_Wn_pfunc   // opt: dmod2 cross-tang
        = NULL,               // shape, [nested]
    DS_pfunc* src2_Wnn_pfunc  // opt: dmod2 curvature
        = NULL,               // shape, [nested]
    void* src1_data           // application data
        = NULL,               // stored for tgt1_dmod
    void* src2_data           // application data
        = NULL,               // stored for tgt2_dmod
    void(* src_CW_func)       // function to
        (void* src_data,      // update
        double s,             // in: curve s param
```

```
                                // value
    double* C,                  // out: surface C=[u,v]
                                // point value
    double* Cs,                 // out: surface dC/ds vec
                                // in surf for s
    double* W,                  // out: image space
                                // W=[x,y,z] point val
    double* Wu,                 // out: image space dW/du
                                // for s
    double* Wv,                 // out: image space dW/dv
                                // for s
                                // tangent for s
    double* Wuu,                // out: image space
                                // d2W/du2
    double* Wuv,                // out: image space
                                // d2W/dudv
    double* Wvv,                // out: image space
                                // d2W/dv2
    double& dist2)              // out: dist**2
    = NULL,                     // (xyz,proj_to_3d(uv))
int behavior                    // in: OR of
    = (1 << 4)|(0)|             // DM_POS_FREE
    (1 << 10)|(0),              // DM_POS_FIXED
                                // DM_POS_LINKED
                                // DM_POS_2_FREE
                                // DM_POS_2_FIXED
                                // DM_POS_2_LINKED
                                // DM_TAN_FREE
                                // DM_TAN_FIXED
                                // DM_TAN_LINKED
                                // DM_TAN_2_FREE
                                // DM_TAN_2_FIXED
                                // DM_TAN_2_LINKED
double tang_gain                // UNUSED
    = 1.0,                      // UNUSED
int tag                         // when = –1 assign next
    = –1,                       // tag number
                                // else use this tag
                                // number
SDM_options* sdmo               // in:SDM_options pointer
    = NULL                      //
);
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description:

Builds and adds to the dmod1 hierarchy a link constraint that will force dmod1 and dmod2 to deform as if they were one, and for their shape to be C0, C1 or discontinuous across this link constraint. The input dmod1 and dmod2 must both be at the root levels of their deformable modeling hierarchies.

When dmod1 and dmod2 come from different deformable modeling hierarchies, i.e. this is the first link constraint between any of the siblings of dmod1 and any of the siblings of dmod2, dmod2 and all of its siblings are added to the sibling list of dmod1. At that time, all of the tag identifiers in dmod2's deformable modeling hierarchy are shifted by the output amount, tag_shift, to guarantee their uniqueness in the combined deformable modeling hierarchy. So, within the dmod2 hierarchy, all tag numbers will be changed as after_tag_number = before_tag_number + tag_shift. When dmod1 and dmod2 are already siblings, tag_shift is set to 0, and all the tag numbers are not changed.

A calling application has to take care to update any data associated by tag values. For the ACIS integration, this means that all ATTRIB_DSMODEL, ATTRIB_CSTRN, and ATTRIB_LOAD derived data associated with the dmod2 hierarchy has to be updated.

The link_constraint is described by the shapes of two curves embedded within the dmods (dmod1 and dmod2) that they constrain. These curves are called curve1 and curve2. The image-spaces of curve1 and curve2 are the original domain-spaces of the DS_dmods that they constrain. Both curve1 and curve2 must share a common parameterization, s. The link constraint forces the points and cross-tangents of common s values on curve1 and curve2 to coincide. Symbolically, we refer to these two curves as C1(s) and C2(s).

The parameterization of curves the C1 and C2 must be controlled, or the link constraints will be artificially stiff. That is the link constraint won't deform as generally as one would like. For a link constraint to work well, the speed of the curves C0 and C1 must be matched, that is the derivatives, dC1/ds and dC2/ds must be related to each other as

dC1(s)/ds = constant * dC2(s)/ds for all values of s.

When the link constraint curves C1 and C2 are just straight lines in the domain spaces of surfaces dmod1 and dmod2, a good strategy for ensuring this relationship is to construct a simple linear curve for C1 and C2 that span the end points of the curve. An example of constructing C1 in this manner is the call DM_make_bspline_curve(rtn_err, 2, 1, 2, 2, knot_index, knot, dof_vec, NULL, 0) ;

Where: int knot_index = {0, 1} ; double knot = {0.0, 1.0} ; double dof_vec = {C0(s_min).u(), C0(s_min).v(), C)(s_max).u(), C0(s_max).v()} ;

The parametric shape of the curves can be given in two ways: either explicitly, by passing DS_pfunc objects (the src1_C_pfunc and src2_C_pfunc input arguments), or procedurally, by passing a callback function (the src_CW_func along with the src1_data and src2_data input arguments) but not by both. That is, either both src1_C_pfunc and src2_C_pfunc are non-NULL or src_CW_func is non-NULL on input, but not both. To apply the link_constraint, point locations for curve1 and curve2 in uv space and xyz space will be evaluated. These computations will be made by either evaluating src1_C_pfunc and src2_C_pfunc directly, or by making a pair of calls on the function, src_CW_func, once with a pointer to src1_data and once with a pointer to src2_data. For ACIS applications the callback function is already written and is called, DS_true_edge_eval and the src_data pointer is populated by the class DS_2acis. Its important to have the callback function or the src_C_pfunc objects generate uv points that are in the original domain space of the deformable models that they constrain (orig_dmod_space). Generating domain locations in either the unit-space or the internal_pfunc_space will cause errors.

When src1_C_pfunc and src2_C_prfunc are supplied, the domain_flag value is used to determine in which space the curves are built. When domain_flag == 0, the src_C_pfuncs' image_spaces are equal to their respective orig_dmod_spaces. When domain_flag == 2, the src_C_pfuncs' image_space are equal to their respective internal_pfunc_spaces. When src_C_pfuncs are supplied in their orig_dmod_spaces, they are scaled by this function from the orig_dmod_space to the internal_pfunc_space and their pointers are stored internally.

The image space shape of the curve constraints will be computed either by projecting the curve's computed uv points into xyz space using the shape definitions for dmod1 and dmod2, or by calling src_CW_func once for curve1 and once for curve2.

When using the src_CW_func callback, applications must store enough additional information for curve1 and curve2 in src1_data and src2_data to be able to compute the curve's uv and xyz point positions given a parametric value, s. The DM Modeler never accesses or frees the src_data memory, but can store and retrieve it for the convenience of the application. These pointers are passed whenever any user written callback functions called.

The behavior of the two curve constraints within the link constraint, called curve1 and curve2, are determined by the value of the behavior argument which is constructed as a combination of the following bit constants. The positions and the cross-tangents along both curve1 and curve2 may be constrained independently in one of three ways: FREE, FIXED, and LINKED.

```
Curve 1 Positions
    DM_POS_FREE
    DM_POS_FIXED
    DM_POS_LINKED

Curve 1 Tangents
    DM_TAN_FREE
    DM_TAN_FIXED
    DM_TAN_LINKED

Curve 2 Positions
    DM_POS_2_FREE
    DM_POS_2_FIXED
    DM_POS_2_LINKED

Curve 2 Tangents
    DM_TAN_2_FREE
    DM_TAN_2_FIXED
    DM_TAN_2_LINKED
```

A FREE curve property is free to move and is not constrained. A FIXED curve property is frozen and will remain in its current shape during subsequent deformations. A LINKED curve is free to move as long as it stays connected to the constraint to which it is linked.

For example.

```
behavior1 = ..... DM_POS_FREE |
                  DM_POS_2_FREE |
                  DM_TAN_FREE |
                  DM_TAN_2_FREE ;
behavior2 = ..... DM_POS_LINKED |
                  DM_POS_2_LINKED |
                  DM_TAN_LINKED |
                  DM_TAN_2_LINKED ;
behavior3 = ..... DM_POS_FIXED |
                  DM_POS_2_LINKED |
                  DM_TAN_LINKED |
                  DM_TAN_2_LINKED ;
behavior4 = ..... DM_POS_FREE |
                  DM_POS_2_FREE |
                  DM_TAN_LINKED |
                  DM_TAN_2_LINKED ;
```

behavior1 specifies a link constraint which is disabled, while behavior2 specifies a C1 link that is free to move in space, and behavior3 specifies a C1 link that is fixed in space but allowed to rotate about its own axis, and behavior4 specifies a link which is discontinuous, but keeps the cross tangents on each side of the link the same.

Mixing a curve1 FREE constraint with a curve2 LINKED is taken to be equivalent to having both curves be LINKED. Mixing a curve1 FIXED constraint with a curve2 LINKED constraint is taken to be equivalent to both curves being FIXED in the same position, which is different than having both curves FIXED, in which case the two curves can be FIXED in different positions.

When the input argument, tag, is –1 the system assigns the newly constructed link a unique tag number, otherwise it assigns it the input tag value. It is the calling application's responsibility to guarantee that assigned tag numbers are unique throughout the entire deformable modeling hierarchy.

Returns the tag id for the newly created link constraint.

Errors: Sets rtn_err to 0 for success, else

DM_parse_tag_flag()
errors,

DM_NULL_INPUT_PTR
when dmod1, dmod2 are NULL on input,

DM_NULL_INPUT_PTR
when src1_C_pfunc, or src2_C_pfunc == NULL on input and
src_WC_func is NULL,

DM_MIXED_CRV_CSTRN_DIM
when the image_dim of
src1_C_pfunc != the dmod1's domain_dim or
src2_C_pfunc != the dmod2's domain_dim,

DM_BAD_CRV_CSTRN_DIM
when either the src1_C_pfunc or src2_C_pfunc are not a curve, i.e. one of their domain_dim's != 1.

DM_BAD_DOMAIN_PT_RANGE
when src1_C_pfunc is not contained by the tag1 dmod or src2_C_pfunc is not contained by the tag2 dmod.

DM_BAD_TAG_VALUE
when the input tag number is not –1 or larger than 0.

DM_NONMONOTONIC_LINK
when the link dcrvs, C1(s) or C2(s) are not described monotonically, that is dC/ds changes sign for some valid values of s.

DM_UNEVEN_LINK_SPEEDS
when the link_cstrn's dcrvs, C1(s0 and C2(s), represented by either src1_C_pfunc and src2_C_pfunc or by the callback src_CW_func, do not have a constantly related speed. That is if the requirement that dC1/ds = constant * dC2/ds is not true for all values of s.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_pt_cstrn

Function: Deformable Modeling, DML Constraints

Action: Adds a point constraint to the target deformable model and returns a new tag identifier or an error.

Prototype:

```
int DM_add_pt_cstrn (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: hierarchy member
    int tag_flag,              // in: specify tgt dmod
                               // 1 = active dmod,
                               // 2 = root dmod,
                               // else = member dmod
                               // with tag=tag_flag.
    DS_dmod* parent_dmod,      // opt: seam's parent
                               // shape or NULL
    int domain_flag,           // specify dpt space
                               // 0=orig_dmod_space,
                               // 1=unit_space,
                               // 2=
                               // internal_pfunc_space.
    double* dpt,               // in: pt's domain loc,
                               // sized:[Domain_dim]
    void* src_data             // opt: carried with
        = NULL,                //  cstrn object
    DS_CSTRN_SRC src_type      // in : cstrn origin.
        = ds_user_cstrn,       // one of:
                               // ds_solid_cstrn,
                               // ds_bound_cstrn,
                               // ds_user_cstrn,
                               // ds_seam_cstrn.
    int behavior               // or of: DM_POS_FIXED
        = (1 << 3),            // DM_TAN_FIXED
                               // DM_CURV_FIXED
                               // DM_TAN_2_FIXED
                               // DM_CURV_2_FIXED
                               // DM_NORM_FIXED
    double* ipt                // in: pt's image pos or
        = NULL,                // NULL to default,
                               // sized:[Image_dim]
    double* domain1_dir        // in : 1st directional
        = NULL,                // derivative direction
                               // or NULL defaults to
                               // proj of tang1_val if
                               // possible else
```

```
                                // u_dir = [1 0].
                                // sized:[Domain_dim]
double* domain2_dir             // in : 2nd directional
    = NULL,                     // derivative direction
                                // or NULL defaults to
                                // proj of tang2_val if
                                // possible else
                                // v_dir = [0 1].
                                // sized:[Domain_dim]
double* tang1_val               // in : dir1 tangent
    = NULL,                     // vector
                                // or NULL to default.
                                // sized:[Image_dim]
double* tang2_val               // in : dir2 tangent
    = NULL,                     // vector or NULL to
                                // default.
                                // sized:[Image_dim]
double* norm_val                // in : normal vector
    = NULL,                     // value or NULL to
                                // default.
                                // sized:[Image_dim]
double* binorm_val              // in : binormal vector
    = NULL,                     // value for curves or
                                // NULL.
                                // sized:[Image_dim]
double* curv1_val               // in : dir1 curvature
    = NULL,                     // value or NULL to
                                // default.
                                // sized:[1]
double* curv2_val               // in : dir2 curvature
    = NULL,                     // value or NULL to
                                // default.
                                // sized:[1]
int tag                         // in: when == –1 assign
    = –1,                       // next tag number
                                // else use this
                                // tag number
SDM_options* sdmo               // in:SDM_options pointer
    = NULL                      // note: sets active dmod
);
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dmapinum.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Builds and adds to the target deformable model a point constraint with a domain point equal to dpt.

The basic flavor for using a point constraint is:

1. Create and apply the constraint to a deformable model, (DM_add_pt_cstrn()).
2. Select which geometry properties are being constrained, (DM_set_cstrn_behavior()).
3. For tracking pt_cstrns, modify the geometry property values by moving the point constraint display points as described above, (DM_set_pt_xyz()).
4. Call solve to see how the constraint modifications changed the deformable model shape, (DM_solve()).
5. Optionally, change the point constraint's domain position and domain directions (directions for directional derivatives on surfaces) with

5a. (DM_set_pt_uv()),

5b. (DM_set_cstrn_pttan_uv_dir()).

6. Optionally, modify the constraint image_pt locations to optimize viewing and/or interactions, DM_set_tan_display_gain() DM_set_comb_graphics().

The point constraint is a device to allow an end–user to specify the differential geometry properties (position, tangent, normal, and curvature) of a point on a deformable model shape. The point constraint works for surfaces, curves in 3D, and curves in 2D. The geometric properties constrained by a point constraint object are called the behaviors of the point constraint. Many but not all combinations of the constraint behaviors are allowed.

A point constraint can control the following geometry properties:

Curves:	Surfaces:
Position	Position
Tangent (velocity)	Tangent1 in domain1_dir direction
Normal direction	Tangent2 in domain2_dir direction
Curvature+Normal+Tangent	Normal direction
Binormal direction	Curvature1+Normal+Tangent1 in domain1_dir direction
	Curvature2+Normal+Tangent2 in domain2_dir direction.

To completely specify a position constraint on a curve or surface, or normal direction constraint behavior on a curve or surface, or a curvature constraint on a curve, or a binormal constraint on a curve, the end–user must supply the following set of information.

1. location on the surface or curve, given by a parametric position,
2. a value for the geometric property being constrained,

2a. a position in image space for the position constraint,

2b. a vector in image space for the normal constraint and a 2nd image space vector specifying the tangent direction,

2c. the curvature value, the normal direction, and the tangent direction,

2d. a vector in image space for the binormal constraint.

The tangent and curvature constraint behaviors for a point constraint on a surface constrain the surface geometric properties in a direction at a point on the surface. A surface can have two independent tangent and/or curvature constraints as long as the domain directions for each constraint is different.

To completely specify a tangent or curvature point constraint behavior on a surface the end–user must supply the following set of information,

Viewing and Modifying the Constraint Parameters

1. location on the surface(domain_pt), given by a parametric position

2a. direction on the surface(domain_dir), given by a parametric vector

3. value for the geometric property being constrained,

3a. for tangent constraints; an image space vector used to specify the image space direction of the tangent.

3b. for curvature constraints; the curvature value, an image space vector used to specify the direction of the surface normal, and a 2nd image space vector used to specify the surface tangent at that point.

In an effort to make the point constraint intuitive to use, the point constraint geometric values may be set indirectly through a set of image points (sometimes called display points). Every geometric property which may be constrained by a point constraint is associated with a display point. To modify the geometric value of a point constraint, such as its position or its tangent direction, move the location of the associated image_pt.

To help support interactive mouse–based editing of the constraint values, the function DM_set_pt_xyz() can be used to directly set the image_pt location or to compute the image point location given an image space line (which can be generated from a viewing pick ray).

These points include:

base_pt = location of the position constraint and base for all vector values.

tang1_pt = tan_display_gain * (base_pt + tang1_val) vector end point used to specify the curve tangent and the surface tangent1 constraint vectors as;

tang2_pt = tan_display_gain * (base_pt + tang2_val) Vector end point used to specify the surface tangent2 vector.

norm_pt = tan_display_gain * norm_size*(base_pt + norm_val) Vector end point used to specify the normal vector direction for a curve or surface. The normal vector sets the normal direction for the normal constraint and the curvature constraints. When a curvature constraint is being used, the norm_pt is not checked for picking or plotting, rather the curv_pts are used. norm_pt is not used by itself it may be used with either or both curv_pt and binorm_pt.

binorm_pt= tan_display_gain * binrom_size*(base_pt + binorm_val) Vector end point used to specify the plane containing the tangent and normal directions for a curve. binorm may be used by itself or with the tangent or with norm_pt.

curv1_pt = C2_display_gain * (base_pt + curv1_val*norm_val) where curv1_val = curvature value for curves or for surface in the domain1_dir direction, norm_val = curve or surface unit–sized normal–vector. When in use, the norm_val vector is always perpendicular to the tang1_val.

curv2_pt = C2_display_gain * (base_pt + curv2_val*norm_val) where curv2_val = curvature value for surfaces in the domain2_dir direction, and norm_val = surface unit–sized normal–vector.

The tan_display_gain and C2_display_gain parameters are included in the point definitions so that the size of the displayed vectors may be modified at run time for viewing convenience.

Input Argument Use:

When domain_flag is set to 1, the input dpt is given in the unit–square space. When domain_flag is set to 0, dpt is given in the orig_dmod_space. When domain_flag is set to 2, dpt is given in the dmod's pfunc's internal_pfunc_space.

The constraint's base_pt location, (the point's location in image space) is set to the ipt value when ipt is not equal to NULL. When ipt equals NULL, the constraint's base_pt image point is set to the result of evaluating the deformable model's shape at the given dpt value. Similarly, the inputs domain1_dir, domain2_dir, tang1_val, tang2_val, norm_val, curv1_val, and curv2_val are all used to set the parameters of the point constraint unless they are NULL (–1.0 for curv1_val and curv2_val), in which case those parameter values default to values found by evaluating the constrained shape.

The default values for domain1_dir and domain2_dir are computed in two steps. If the associated tang_val vector is given then the domain_dir direction is found by projecting that vector into the domain space of the constrained shape. When the associated tang_val is also set to NULL then domain1_dir defaults to the u_direction = [1 0], and the domain2_dir defaults to the v_direction = [0 1]. For curves domain1_dir is always equal to the +s direction and the arrays domain1_dir and domain2_dir are not used.

The bits in behavior determine what properties are constrained at the given domain point. When the DM_POS_FIXED bit is set, the position of the point is constrained. When the DM_TAN_FIXED bit is set, the tangent at the point is constrained. When the DM_CURV_FIXED bit is set, the DM_TAN_FIXED and the DM_NORM_FIXED bits are set by the system and the curvature at the point is constrained. The legal values for behavior include:

For points on curves:

– DM_POS_FIXED,
– DM_TAN_FIXED,
– DM_NORM_FIXED,
– DM_BINORM_FIXED,
– DM_CURV_FIXED.

For points on surfaces:

– DM_POS_FIXED,
– DM_TAN_FIXED,
– DM_TAN_2_FIXED,
– DM_NORM_FIXED,
– DM_CURV_FIXED,
– DM_CURV_2_FIXED.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag).

Returns a tag number for the newly added constraint point.

Errors: DM_parse_tag_flag() errors

DM_scale_unit_dpt_to_pfunc() errors

DM_NO_ROOT_DMOD
The the input deformable model cannot be NULL.

DM_NULL_INPUT_PTR
The input dpt value cannot be NULL.

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model.

DM_BAD_TAG_VALUE
The tag value must be –1, to have the system assign one, or 2 or greater.

DM_BAD_SRC_TYPE_PARENT
When src_type == ds_seam_cstrn, parent_dmod must be non–NULL, else parent_dmod must be NULL.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_pt_press

Function: Deformable Modeling, DML Loads

Action: Adds a pressure point to deformable model and returns a new tag identifier or an error.

Prototype:

```
int DM_add_pt_press (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: member of target
                               // dmod hierarchy to
                               // search
    int tag_flag,              // in: specify tgt dmod
                               // 1 = active dmod
                               // 2 = root dmod
                               // else = member dmod
                               // with tag=tag_flag
    int domain_flag,           // orig_dmod_space,
                               // 1=dpt in unit_space,
                               // 2=dpt in
                               // internal_pfunc_space.
    double* dpt,               // in: load's domain
                               // loc, [u,v]
                               // sized:[Domain_dim]
    double gain                // in: magnitude of
        = 0.0,                 // load gain
    int negate_flag            // in: change normal dir
        = 0,                   // (1=negate,0=do not)
    int tag                    // in: when = –1 assign
        = –1,                  // next tag number
                               // else use this
                               // tag number
    SDM_options* sdmo          // in:SDM_options pointer
        = NULL                 // note: sets active dmod
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Modifies the target deformable model.

Builds and adds to the target deformable model a DS_pressure point with a domain point and gain as specified by the input arguments. When tag equals –1 the system assigns the next available tag number to the newly created load. Otherwise, the input tag value is used as the load's tag number. When the negate_flag equals 0 the pressure load is applied in the deformable model's normal direction. When negate_flag is set, the pressure vector is negated before being applied.

When domain_flag is set to 1, dpt is given in the unit range and mapped to the dmod's actual domain range by this function. When domain_flag is set to 0, dpt is given in the dmod's original domain range (the one with which it was created). When domain_flag is set to 2, dpt is given in the dmod's pfunc's internal_pfunc_space.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag)

Returns a newly created DS_load's tag number.

Errors: DM_parse_tag_flag() errors

DM_scale_unit_dpt_to_pfunc() errors

DM_BAD_TAG_VALUE
The tag value must be –1, to have the system assign one, or 2 or greater.

DM_NO_ROOT_DMOD
The the input deformable model cannot be NULL.

DM_BAD_NEGATE_FLAG_VALUE
The negate_flag must be a 0 or a 1.

DM_NULL_INPUT_PTR
The dpt cannot be NULL on input.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_spring

Function: Deformable Modeling, DML Loads

Action: Adds a spring to a deformable model and returns a new tag identifier or an error.

Prototype:

```
int DM_add_spring (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: member of target
                            // dmod hierarchy to
                            // search
    int tag_flag,           // in: specify tgt dmod
                            // 1 = active dmod
                            // 2 = root dmod
                            // else = member dmod
                            // with tag=tag_flag
    int domain_flag,        // orig_dmod_space,
                            // 1=dpt in unit_space,
                            // 2=dpt in
                            // internal_pfunc_space.
    double* dpt,            // in: domain_pt loc of
                            // surf spring end
                            // sized:[domain_dim]
                            // ptr:[not-nested]
    double* ipt             // in: image_pt loc of
        = NULL,             // free spring end
                            // sized:[image_dim]
                            // ptr:[not-nested]
    int ipt_flag            // in: 0: let free_pt =
        = 0,                // image_loc of dpt
                            // 1: let free_pt = ipt
    double gain             // in: magnitude of
        = 0.0,              // load gain
    int tag                 // in: when = -1 assign
        = -1,               // else use this
                            // tag number
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              // note: sets active dmod
);
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description:

Builds and adds to the target deformable model a DS_spring with its domain point, image point, and gain specified by the input arguments. A spring is a force that acts on the surface at the point where the spring is attached, in a direction pointing to the spring's image point. gain is the spring's stiffness. The spring's image point can be specified in two ways: explicitly, by setting ipt_flag to 1 and giving an image point vector *x*,*y*,*z*, or by setting ipt_flag to 0 and having the system take the initial image point location from the deformable model's current location for the domain point's dpt position.

When domain_flag is set to 1, dpt is given in the unit range and mapped to the pfunc's actual domain range by this function. When domain_flag is set to 0, dpt is given in the dmod's original domain range. When domain_flag is set to 2, dpt is given in the dmod's pfunc's internal domain range.

When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag)

Returns a new load's tag number when successful.

Errors:

DM_parse_tag_flag() errors

DM_BAD_IPT_FLAG_VALUE
The ipt_flag must be 0 or 1.

DM_NULL_INPUT_PTR
The input ipt value cannot be NULL and the ipt_flag cannot be 1.

DM_NULL_INPUT_PTR
The dpt input pointer cannot be NULL.

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_spring_set

Function: Deformable Modeling, DML Loads

Action: Adds a set of springs to a deformable model and returns a new tag identifier or an error.

Prototype:

```
int DM_add_spring_set (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: member of target
                               // dmod hierarchy to
                               // search
    int tag_flag,              // in: specify tgt dmod
                               // 1 = active dmod
                               // 2 = root dmod
                               // else = member dmod
                               // with tag=tag_flag
    int domain_flag,           // in: 0=domain_pts in
                               // orig_dmod_space,
                               // 1=domain_pts in
                               // unit_space,
                               // 2=domain_pts in
                               // internal_pfunc_space
    int pt_count,              // in: number of springs
                               // in spring_set
    double* domain_pts,        // i/o: Spring surf
                               // end pts
                               // 1d=[u0,u1,...un],
                               // 2d=[uv0,...,uvN]
                               // Sized:[pt_count*
                               // domain_dim]
                               // ptr:[not-nested]
    double* free_pts           // in: Spring free end
        = NULL,                // pts or NULL
                               // [xyz0,...,xyzN]
                               // Sized:[pt_count*
                               // image_dim]
                               // ptr:[not-nested]
    double gain                // in: stiffness of all
        = 0.0,                 // springs
    int tag                    // next tag number
        = -1,                  // else use this
                               // tag number
    SDM_options* sdmo          // in:SDM_options pointer
        = NULL                 // note: sets active dmod
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Builds and adds to the target deformable model a DS_spring_set using domain_pts and free_pts and the gain input arguments. A DS_spring_set is a set of springs that all share a common gain. When free_pts is NULL each spring's free_pt position is calculated using the spring's domain_pt location and the current shape of the deformable model.

When domain_flag is set to 1, every dpt is given in the unit range and mapped to the dmod's actual domain range by this function. When domain_flag is set to 0, the dpts are given in the dmod's original domain range. When domain_flag is set to 2, the dpts are given in the dmod's pfunc's internal domain range.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag)

Returns a newly created DS_load's tag number.

Errors: DM_parse_tag_flag() errors

DM_scale_unit_dpt_to_pfunc() errors

DM_BAD_TAG_VALUE
The input tag value must be –1 or positive.

DM_NO_ROOT_DMOD
The the input deformable model cannot be NULL.

DM_NULL_INPUT_PTR
The dpt input pointer cannot be NULL.

DM_BAD_PT_COUNT_VALUE
The pt_count cannot be equal to or less than 0.

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_surface_patch

Function: Deformable Modeling, DML Patches

Action: Creates and adds a patch to the patch hierarchy and returns a new patch tag identifier or an error.

Prototype:

```
int DM_add_surface_patch (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: dmod to be the
                               // parent
    int domain_flag,           // in: 0=domain_pts and
                               // seams in
                               // orig_dmod_space,
                               // 1=domain_pts and seams
                               // in unit_space,
                               // 2=domain_pts and seams
                               // in
                               // internal_pfunc_space
    double pt1[2],             // in: min-corner of
                               // domain box bounding
                               // all seams
    double pt2[2],             // in: max-corner of
                               // all seams
    int refinement,            // in: parent knot
                               // spacing/child knot
                               // spacing
    void* patch_entity,        // in: app entity ptr
                               // stored with patch
    int seam_count,            // in: number of domain
                               // curves in seam
                               // (may be 0)
    DS_pfunc** seam,           // in: seam curves
                               // sized:[seam_count]
    void** seam_data,          // in: app data stored
                               // with each seam
                               // sized:[seam_count]
    SDM_options* sdmo          // in:SDM_options pointer
        = NULL                 //
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description:

Adds a patch to the input deformable model. The domain of the patch is a rectangle connected to its parent by a seam. The size of the patch domain is given by its minimum and maximum corner points, pt1 and pt2. The shape of the seam is given by a set of domain (2D) curves that all must lie within the given patch domain. Think of the patch domain as a bounding box for all the seam curves. (The convenience methods, DM_build_square_seam() and DM_build_ellipse_seam() can be used to build the input arguments, seam_count and seam, for either a square or an elliptical patch seam boundary and will properly assign tag numbers for each of the seam entities).

Domain pts, pt1 and pt2, and the image of the seam curves can be given in either the unit_space, the orig_dmod_space, or the internal_pfunc_space. When domain_flag is set to 0, these values are expected to be in the orig_dmod_space. When domain_flag is set to 1, these values are expected to be in the unit_space. When domain_flag is set to 2, these values are expected to be in the internal_pfunc_space. The orig_dmod_space is the domain range equal to the pfunc's domain range at the time the dmod was constructed. The internal_pfunc_space is the domain range being used internally by the dmod's pfunc. This range may vary as the deformable modeling package adjusts the range to maximize the numerical accuracy of computing high order derivatives.

The refinement argument specifies the number of knots in the child patch compared to the parent patch. For example, a refinement value of 2 will double the density of knots in the child compared to the parent. refinement must be an integer value greater than 0.

The patch_entity pointer is stored with the patch and can be accessed by an application with the method DS_dmod–>Entity() and may be changed with the method DS_dmod–>Set_entity(). The seam_data array is a set of pointers, each stored with its corresponding seam that can be accessed by an application by using, DS_dmod–>Seam(ii)–>Src_data() and may be set by the call DS_dmod–>Seam(ii)–>Set_src_data().

When using hierarchical surface patches you will find that if the degree of the surface is less than that used for the patch, deformations will look unnatural. This is especially true if C1 seam constraints are being used. This is because the patch doesn't have enough flexibility to conform to the seam constraints and deform as well. Increasing the surface degree adds much more flexibility to the surfaces. Certainly, degree 4 and 5 surfaces work better than degree 3 surfaces.

Returns the tag of the newly created patch or an error.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_scale_unit_dpt_to_pfunc() errors

DM_UNCONTAINED_CHILD
The input patch boundary must be in the range of 0 to 1.

DM_BAD_REFINEMENT_VALUE
The input refinement must be larger than 0.

DM_PATCH_OCCLUDES_ROOT
The input patch completely covers the deformable model.

DM_BAD_DOMAIN_PT_RANGE
The seams must be completely contained by the new child patch.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_add_vector_load

Function: Deformable Modeling, DML Loads

Action: Adds a vector load to deformable model and returns a tag identifier or an error.

Prototype:

```
int DM_add_vector_load (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: member of target
                               // dmod hierarchy to
                               // search
    int tag_flag,              // in: specify tgt dmod
                               // 1 = active dmod
                               // 2 = root dmod
                               // else = member dmod
                               // with tag=tag_flag
    double* image_vec          // in: direction of
        = NULL,                // vector_load or NULL
                               // sized:[image_dim]
                               // [not–nested]
    double gain                // in: magnitude of
        = 0.0,                 // load gain
    int tag                    // in: when = –1 assign
        = –1,                  // next tag number
                               // else use this
                               // tag number
    SDM_options* sdmo          // in:SDM_options pointer
        = NULL                 // note: sets active dmod
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description:

Builds and adds to the target deformable model a vector load specified by an image space vector and a gain. The direction of the vector load is specified either by image_vec, or computed as the shape normal at the center of the deformable model when image_vec is NULL.

A vector load is a constant vector force applied to the entire deformable shape. This load can be used to simulate a gravitational effect. The direction of the vector load is the direction of the input image_vec. Its magnitude is stored in gain.

tag_flag selects the target deformable model. When tag equals –1 the next available tag number is assigned to the newly created load. Otherwise, the input tag number is used. Other valid values for the tag flag are:

1 = active deformable model
2 = root deformable model

Otherwise, the target is the deformable model whose tag identifier equals abs(tag_flag)

Returns a new load's tag number when successful.

Errors: DM_parse_tag_flag() errors

DM_BAD_TAG_VALUE
The input tag value must be –1 or positive.

DM_NO_ROOT_DMOD
The the input deformable model cannot be NULL.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_assign_next_tag

Function: Deformable Modeling, DML Tags

Action: Increments and returns the tag count and the next available tag number or an error.

Prototype:
```
int DM_assign_next_tag (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: dmod member of
                            // a hierarchy/ tree
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Increments and returns the value of dmo_tag_count. Use this value as the tag value for the next tag object being created. If you always use this function to select the tag value for each tag object entered into a deformable model tag hierarchy, then every tag object in the hierarchy will have a unique value.

Returns a positive tag identifier number.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: System routine

DM_build_ellipse_seam

Function: Deformable Modeling, DML Patches

Action: Builds a curve set to act as a seam.

Prototype:

```
void DM_build_ellipse_seam (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: dmod for new seam
                            // (the parent)
    int domain_flag,        // in: 0=domain_pts in
                            // orig_dmod_space,
                            // 1=domain_pts in
                            // unit_space,
                            // 2=domain_pts in
                            // internal_pfunc_space.
    double cpt[2],          // in: center point for
                            // patch
    double axis1[2],        // in: vector from cpt
                            // to 1st axis end point
    double axis2[2],        // in: vector from cpt
                            // to 2nd axis end point
    double min[2],          // out: lower domain
                            // ellipse bound, output
                            // in orig_dmod_space
    double max[2],          // out: upper domain
                            // ellipse bound, output
                            // in orig_dmod_space
    int& pfunc_count,       // out: always set to 1
    DS_pfunc**& pfunc,      // out: newly created
                            // pfunc array (NULL on
                            // input), output in
                            // orig_dmod_space,
                            // mallocs: pfunc array
                            // and each pfunc entry
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”dshusk/dskernel/dmapi.hxx”
#include ”dshusk/dskernel/dsdmod.hxx”
#include ”dshusk/dskernel/dspfunc.hxx”
#include ”dshusk/dskernel/sdm_options.hxx”
```

Description: Builds and returns an elliptical curve suitable for use as a patch seam boundary in the target deformable model. Checks that pfunc is NULL on entry to the subroutine. Builds a DS_circ object using the cpt, axis1, and axis2 positions scaled to the target deformable model's domain space and then computes and stores the minimum sized bounding box for the ellipse in 0 to 1 space. This function checks that the minimum and maximum points lie within the target deformable model's object's domain space. When the check fails, pfunc_count is set to 0 and pfunc is left unaltered. When the check passes, this function stores in pfunc the DS_circular curve that forms the ellipse and sets the pfunc_count value to 1.

The newly constructed seams are built in an image–space that is equal to the orig_dmod_space of the input dmod. The min and max domain point locations are scaled into this orig_dmod_space. These seams and min and max domain point values can be used directly as input to the function DM_add_surface_patch as long as the domain_flag value for the DM_add_surface_patch() call is set to 0.

Verifies that minimum and maximum represent legitimate points contained within the domain space of the this object.

When domain_flag is set to 1, cpt, axis1, and axis2 are given in the unit range and mapped to the dmod's orig_dmod_space domain range by this function. When domain_flag is set to 0, these inputs are given in orig_dmod_space. When domain_flag is set to 2, these inputs are given in internal_pfunc_space.

mallocs the pfunc array and the memory for DS_circ member of the array. The calling program is responsible for freeing this memory.

Errors: DM_parse_tag_flag() errors

DM_BAD_DOMAIN_PT_RANGE
The minimum and maximum must be within the range of 0 to 1.

DM_NON_NULL_OUTPUT_PTR
pfunc must be NULL on entry.

DM_ZERO_AREA_PATCH
The requested child patch must not have a zero area.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_build_fillet_square_seam

Function: Deformable Modeling, DML Patches

Action: Builds a curve set to act as a seam.

Prototype:

```
void DM_build_fillet_square_seam (
    int& rtn_err,           // out: 0=success or
                            // neg err code
    DS_dmod* dmod,          // n: dmod for new seam
                            // (the parent)
    int domain_flag,        // in : 0=min, max,
                            // radius in
                            // orig_dmod_space,
                            // 1=min, max, radius in
                            // unit_space, 2=min,
                            // max, radius in
                            // internal_pfunc_space.
    double min[2],          // i/o: patch lower
                            // domain point, output
                            // in orig_dmod_space
    double max[2],          // i/o: patch upper
                            // domain point, output
                            // in orig_dmod_space
    double radius,          // patch corner fillet
                            // radius
    int& pfunc_count,       // number of pfuncs
                            // in pfunc array
    DS_pfunc**& pfunc,      // newly created
                            // pfunc array (NULL on
                            // input) output in
                            // orig_dmod_space
                            // mallocs: pfunc
                            // array and each pfunc
                            // entry
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Modifies pfunc_count, pfunc, active_dmod.

This function builds and returns a set of curves suitable for use as a patch seam boundary in the tgt_dmod. Checks that pfunc is NULL on entry to the subroutine and that the min and max points lie within the tgt_dmod's domain space. When the check fails, pfunc_count is set to 0 and pfunc is left unaltered. When the check passes, builds and stores in pfunc an array of 4 curves that form a square in the domain space of the tgt_dmod. Sets the pfunc_count value to 4.

The newly constructed seams are built in an image–space that is equal to the orig_dmod_space of the input dmod. The min and max domain point locations are scaled into this orig_dmod_space. These seams and min and max domain point values can be used directly as input to the function DM_add_surface_patch as long as the domain_flag value for the DM_add_surface_patch() call is set to 0.

Verifies that minimum and maximum represent legitimate points contained within the domain space of the this object.

When domain_flag is set to 1, min, max, and radius are given in the unit range and mapped to the dmod's orig_dmod_space domain range by this function. When domain_flag is set to 0, these inputs are given in orig_dmod_space. When domain_flag is set to 1, these inputs are given in the internal_pfunc_space. It mallocs the pfunc array and the memory for each member in the array. The calling program is responsible for freeing this memory.

Errors: Sets rtn_err to
0 for success, else

DM_parse_tag_flag()
errors,

DM_BAD_DOMAIN_PT_RANGE
when min or max are not in the range of 0 to 1 or within the tgt_dmod.

DM_NON_NULL_OUTPUT_PTR
when pfunc is not NULL on entry.

DM_FILLET_RADIUS_TOO_BIG
when fillet radius is larger than Min(max[0]–min[0],max[1]–min[1])/2.0

DM_FILLET_RADIUS_NOT_POS
when fillet radius is negative or zero.

DM_ZERO_AREA_PATCH
The requested child patch must not have a zero area.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_build_square_seam

Function: Deformable Modeling, DML Patches

Action: Builds a curve set to act as a child patch's seam given the parent's deformable model.

Prototype:

```
void DM_build_square_seam (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: dmod for new seam
                            // (the parent)
    int domain_flag,        // // in: 0=min and max
                            // in orig_dmod_space,
                            // 1=min and max in
                            // unit_space,
                            // 2=min and max in
                            // internal_pfunc_space
    double min[2],          // i/o: lower domain
                            // point output in
                            // orig_dmod_space
    double max[2],          // i/o: upper domain
                            // point output in
                            // orig_dmod_space
    int& pfunc_count,       // out: number of pfuncs
                            // in pfunc array
    DS_pfunc**& pfunc,      // out: newly created
                            // pfunc array (NULL on
                            // input) output in
                            // orig_dmod_space.
                            // mallocs: pfunc array
                            // and each pfunc entry
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description:

Modifies pfunc_count, pfunc, the active deformable model.

Builds and returns a set of curves suitable for use as a patch seam boundary in the target deformable model. Checks that pfunc is NULL on entry to the subroutine and that the minimum and maximum points lie within the target deformable model's domain space. When the check fails, pfunc_count is set to 0 and pfunc is left unaltered. When the check passes, this function builds and stores in pfunc an array of four curves that form a square in the domain space of the target deformable model. It also sets the pfunc_count value to 4.

The newly constructed seams are built in an image–space that is equal to the orig_dmod_space of the input dmod. The min and max domain point locations are scaled into this orig_dmod_space. These seams and min and max domain point values can be used directly as input to the function DM_add_surface_patch as long as the domain_flag value for the DM_add_surface_ptach() call is set to 0.

Verifies that minimum and maximum represent legitimate points contained within the domain space of the this object.

When domain_flag is set to 1, dpt is given in the unit range and mapped to the pfunc's actual domain range by this function. Otherwise, dpt is given in the pfunc's actual domain range.

mallocs the pfunc array and the memory for each member in the array. The calling program is responsible for freeing this memory.

Errors: DM_parse_tag_flag() errors

DM_BAD_DOMAIN_PT_RANGE
The minimum and maximum must be within the range of 0 to 1.

DM_NON_NULL_OUTPUT_PTR
pfunc must be NULL on entry.

DM_ZERO_AREA_PATCH
The requested child patch must not have a zero area.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_build_square_zone

Function: Deformable Modeling, DML Patches

Action: Builds and returns a DS_zone object suitable for use as an area constraint.

Prototype:

```
DS_zone* DM_build_square_zone (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: target dmod
                            // for new zone
    int domain_flag,        // in: specify coordinate
                            // system for min and max
                            // pts.
                            // 0=orig_dmod_space,
                            // 1=unit_space,
                            // 2=
                            // internal_pfunc_space.
    double min[2],          // in : square's
                            // lower–left pt
    double max[2],          // in : square's
                            // upper–right pt
                            // mallocs: returned
                            // zone object
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dszone.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description:

Builds and returns a DS_zone object suitable for use as input to the DM_add_area_cstrn() call. A DS_zone object specifies a sub–region within the deformable model's DS_pfunc object.

The input arguments min and max specify diagonally opposite corners of a rectangular subdomain of the DS_pfunc. This subdomain specifies the DS_zone sub–region, via the underlying DS_pfunc surface.

The inputs min and max are domain points which can be given in one of three different coordinate systems. The input domain flag value specifies which coordinate system is being used for these points.

When domain_flag = 0, the domain points are given in the original_dmod_space, (the domain range of the DS_pfunc object at the time it was used to build the DS_dmod object and before it may have been scaled internally.)

When domain_flag = 1, the domain points are given in the unit space, a square region that ranges from 0.0 to 1.0.

When domain_flag = 2, the domain points are given in the internal_pfunc_space, (the domain range currently stored in the deformable model's DS_pfunc object. This range often varies from the original_dmod_space due to internal scalings of the DS_pfunc's domain.)

The internal representation of shape stored within the returned zone is saved in the internal_pfunc_space coordinate system.

mallocs the returned zone object is malloced and must be freed by the application at the appropriate time. However, once the zone is used to make an area constraint or some other kind of tag object, the DM library will free this memory when asked to free the tag object.

Errors: DM_parse_tag_flag() errors

DM_NULL_INPUT_PTR
When dmod is not NULL on input.

DM_BAD_DOMAIN_FLAG_VALUE
When the domain_flag value is not one of, 0=orig_dmod_space, 1=unit-space, 2=internal_pfunc_space.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_classify_tag

Function: Deformable Modeling, DML Tags

Action: Checks the deformable model tree for a tag.

Prototype:

```
DS_TAGS DM_classify_tag (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: member of target
                            // dmod hierarchy to
                            // search
    int tag,                // in: src tag number
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              // note: sets active dmod
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dmapinum.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Searches all the patch deformable models, constraints, loads, and control points for one that has the input tag value. When the tag value is found it returns the DS_tag_type enumeration value for its type and sets the active deformable model pointer to be the deformable model owner of the identified tag object. When not found it returns DS_tag_none

Returns the DS_tag_type for the tag or DS_tag_none. Type values are:

DS_tag_none	= Value for no such tag object
DS_tag_control_pt	= Surface control points
DS_tag_pt_press	= Point pressures
DS_tag_dist_press	= Distributed pressure
DS_tag_spring	= Spring load
DS_tag_spring_set	= Ordered point set load
DS_tag_crv_load	= Curve spring load
DS_tag_dyn_load	= Dynamic load
DS_tag_pt_cstrn	= Point constraint
DS_tag_crv_seam [Base DS_cstrn]	= Seam point constraint
DS_tag_crv_cstrn [Base DS_cstrn]	= Curve constraint
DS_tag_crv_seam [Base DS_cstrn]	= Seam curve constraint
DS_tag_srf_dmod [Base DS_dmod]	= Surface deformable model
DS_tag_crv_dmod [Base DS_dmod]	= Curve deformable model
DS_tag_vector_load [Base DS_load]	= Vector load
DS_tag_attractor [Base DS_load]	= Attractor load
DS_tag_link_cstrn [Base DS_cstrn]	= Multi–surf link constraint
DS_tag_area_cstrn [Base DS_cstrn]	= Isolated deformation region
DS_tag_link_load [Base DS_cstrn]	= Multi–surf link load
DS_tag_track_pos_crv [Base DS_dmod]	= tracking position curve dmod
DS_tag_track_tan_crv [Base DS_dmod]	= tracking tangent curve dmod
DS_tag_track_curv_crv [Base DS_dmod]	= tracking curvature curve dmod

Errors: DM_NON_NULL_OUTPUT_PTR
pfunc must be NULL on entry.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_convert_loads_and_cstrns

Function: Deformable Modeling, DML Constraints, DML Loads

Action: Converts curve objects into load objects and vice versa and returns a new tag identifier or an error.

Prototype:
```
int DM_convert_loads_and_cstrns (
    int& rtn_err,              // out: 0=success or neg
                               // err code
    DS_dmod* dmod,             // in: member of target
                               // dmod hierarchy to
                               // search
    int tag,                   // in: load/cstrn
                               // identifier note:
    DM_target_memory flag      // in: one of
    = DM_remember_target,      // DM_remember_target
                               // (default)
                               // DM_forget_target
    SDM_options* sdmo          // in:SDM_options pointer
        = NULL                 // note: sets active dmod
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/dmapinum.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: This function replaces DM_convert_cstrns_and_loads. The old functionality is now supported by calling this function with the non-default argument DM_forget_target. This was necessary due to a behavior change in converting between curve loads and curve constraints; the new function remembers target information.

This function modifies the tag object's containing deformable model. When allowed, it builds a load from the input constraint and then deletes the constraint (or vice versa). When operating on user-defined curve constraints it converts the constraint into a load, keeping the tag number the same.

The current shape of a curve object is defined as the projection of its domain curve into image space using the current deformable surface shape. The mem_flag argument specifies whether the new curve object will remember the old curve objects target shape. If DM_forget_target is specified, the new curve object will use its current shape as a target. If DM_remember_target is specified, it will use the old object's target shape.

For non-user curve objects, when the curve constraint can be disabled, it is disabled and the tag of the newly created curve load is returned. When the curve constraint cannot be deleted or disabled, no curve load is created and the value DM_UNSTOPABLE_CSTRN is returned. This will be the case for solid and seam constraints.

Note that link constraints and link loads cannot be converted, since link constraints are being deprecated.

If tag is a constraint that is:

- not stoppable or the src_type is not ds_user_cstrn, then return is DM_NOT_USER_CSTR.
- a curve and the src_type is ds_boundary_cstrn, then add a curve load. The behaviors (including target curves) remain the same if DM_remember_target is passed in, otherwise behaviors remain the same but the target becomes the spline curve approximation to the current shape of the curve constraint.
- a curve and the src_type is ds_user_cstrn, then convert the curve constraint into a curve load. The behaviors (including target curves) remain the same if DM_remember_target is passed in, otherwise behaviors remain the same but the target becomes the spline curve approximation to the current shape (defined above) of the curve constraint. Note that a new tag object is not created; the returned tag ID will be the same as the tag argument passed in.
- a point, then add a spring.
- deletable (i.e. user constraints), then remove or delete the constraint.
- stoppable (i.e. boundary constraints), then turn it off.

If tag is a load that is:

– a curve, then convert the curve load into a curve constraint. The behaviors (including target curves) remain the same if DM_remember_target is passed in, otherwise behaviors remain the same but the target becomes the current shape of the curve load.
– A pressure point or a spring, then add one point constraint and remove and delete the load.
– A spring_set, add one point for each spring and remove and delete the load and return the last tag added.
– Another kind of load, then return DM_LOAD_NOT_CONVERTIBLE.

Returns a newly created tag object's tag identifier for success.

Errors:

DM_NO_ROOT_DMOD
The the input deformable model cannot be NULL.

DM_TAG_OBJECT_NOT_FOUND
when the input tag value fails to identify a load or constraint.

DM_UNSTOPABLE_CSTRN
The identified constraint state is set to unstoppable.

DM_NOT_USER_CSTRN
The identified constraint type must be a ds_user_cstrn or a ds_bound_cstrn)

DM_LOAD_NOT_CONVERTIBLE
The identified load must be a DS_crv_load, a DS_spring, or a DS_spring_set.

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model's pfunc.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_copy_dmod

Function: Deformable Modeling, DML Create and Query, Deformable Model Management

Action: Creates a deep copy of the input deformable model and returns a pointer to new deformable model object.

Prototype:
```
DS_dmod* DM_copy_dmod (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: dmod object
                            // to copy
    int walk_flag           // in: 0='this' only,
        = 0,                // 1='this' and offspring
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Makes and returns a pointer to a deep copy of the input deformable model.

NULL for failure or a pointer to an independent copy of of the input deformable model object.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_BAD_WALK_FLAG_VALUE
The walk_flag must be 0, 1 or 2.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: System routine

DM_copy_pfunc

Function: Deformable Modeling, DML Create and Query

Action: Creates a deep copy of the input pfunc and returns a pointer to the new DS_pfunc object.

Prototype:
```
DS_pfunc* DM_copy_pfunc (
    int& rtn_err,           // out: 0= success
                            // or DM_NULL_INPUT_PTR
    DS_pfunc* pfunc,        // in: pfunc object to
                            // copy, [not-nested]
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Makes and returns a pointer to a deep copy of the input pfunc.

Never use one pfunc to create two deformable model objects. Always use a separate copy of a pfunc for each deformable model.

Returns a NULL for failure or a pointer to an independent copy of of the input pfunc object.

Errors: DM_NULL_INPUT_PTR
The pfunc cannot be NULL on entry.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: System routine

DM_delete_dmod

Function: Deformable Modeling, DML Create and Query

Action: Clears the deformable model's data structure.

Prototype:
```
void DM_delete_dmod (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod*& dmod,         // i/o: ptr to valid
                            // dmod object set
                            // to NULL on exit
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Frees and clears all data values associated with the underlying deformable model and sets the deformable model pointer to NULL. Recursively deletes all of the dmod's children and all of the dmod's younger siblings.

1. The deformable model pointer is invalid after this call.
2. The pfunc object referenced by this deformable model is also deleted. Do not call DM_delete_pfunc on that object.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: System routine

DM_delete_pfunc

Function: Deformable Modeling, DML Create and Query

Action: Clears the pfunc data structure.

Prototype:
```
void DM_delete_pfunc (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_pfunc*& pfunc,       // i/o: ptr to valid
                            // pfunc object set to
                            // NULL on exit
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Frees and clears all data values associated with the input DS_pfunc and sets the pfunc pointer to NULL.

1. The DS_pfunc pointer is invalid after this call.
2. Do not use this function to delete the pfunc pointer after it has been used to build a DS_dmod deformable modeling object. The pfunc pointer will be deleted when the DS_dmod pointer is deleted.

Errors: DM_NULL_INPUT_PTR
The pfunc cannot be NULL on entry.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: System routine

DM_delete_zone

Function: Deformable Modeling, DML Create and Query

Action: Clears the zone data structure.

Prototype:
```
void DM_delete_zone (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_zone*& zone,         // i/o: ptr to valid
                            // zone object set to
                            // NULL on exit
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dszone.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Frees and clears all data values associated with the input DS_zone object and sets the zone pointer to NULL.

The DS_zone pointer is invalid after this call.

Do not call this function on a zone that has been used to create a tag object, e.g., an area constraint. That zone will be freed when the area constraint is freed.

Errors: DM_NULL_INPUT_PTR
The zone cannot be NULL on entry.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_draw_icon

Function: Deformable Modeling

Action: Broadcast draw method. Draw these icons using the data encapsulated by the draw_args.

Prototype:

```
void DM_draw_icon (
    int& rtn_err,            // error code
    const DM_icon_draw_args& args,// args passed
    DS_dmod* dmod,           // model to query
    const int* tags,         // tag object owning
                             // query
    int ntags,               // number of tags
    SDM_options* sdmo        // in:SDM_options pointer
        = NULL               //
    );

void DM_draw_icon (
    int& rtn_err,            // error code
    const DM_icon_draw_args& args,// args passed
    DS_dmod* dmod,           // model to query
    int tag,                 // tag object owning
                             // icon
    SDM_options* sdmo        // in:SDM_options pointer
        = NULL               //
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dm_icon_args.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: This function calls the icon's Draw method. The data encapsulated by the draw_args is passed through this method. rtn_err will return 0 for success, or a negative error code.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_TAG_OBJECT_NOT_FOUND
Could not find tag object with the given tag.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_elevate_dmod_degree

Function: Deformable Modeling, Deformable Model Management

Action: Increases the deformable model's polynomial degree by adding more degrees of freedom per element.

Prototype:
```
void DM_elevate_dmod_degree (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: the target dmod
                            // to modify
    int cont_flag,          // in: 0=preserve current
                            // elem continuity
                            // 1=increment element
                            // continuity note:
                            // preserving elem
                            // continuity allows
                            // the new shape to be
                            // exactly the same as
                            // the original shape.
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Increases the basis function's polynomial degree for B–spline and NURB deformable models. This is a recursive function and all of the deformable model's offspring are elevated as well. This acts to add new dofs for the deformable model but does not increase the number of unique knots. The cont_flag argument determines how the element-to-element continuity behavior changes when the deformable model's degree is elevated.

When the cont_flag is set to 1, the multiplicity counts on all the internal knots are left as they are and the continuity across the knots increases by one degree. To accommodate this one more degree of freedom (or control point) is added to each axis. So when a 6x6 control point surface has its degree elevated with cont_flag equal to 1, it becomes a 7x7 control point surface. The down side is that the elevated deformable model shape will not be able to exactly assume the shape of the original surface.

When cont_flag is set to 0, the multiplicity count for all internal knots is increased by one, keeping the original element–to–element continuity while increasing the degree of the polynomial within each element. To accommodate this, one control point is added for each element along each axis. So a 6x6 control point surface made of an array of 3x3 bicubic elements will become a 9x9 control point surface. The upside, bought by the addition of all these control points, is that the elevated surface can exactly assume the shape of the original surface.

All the deformable modeling algorithms are a function of the degree of freedom count. If you can possibly live with using cont_flag equals 1 you should do so.

When using hierarchical surface patches you will find that if the degree of the surface is less than 4 the patch deformations will look unnatural. This is especially true if C1 seam constraints are being used. This is because the patch just doesn't have enough flexibility to conform to the seam constraints and to deform as well. Increasing the surface degree adds a lot of flexibility to the surfaces. Certainly degree 4 and 5 surfaces work better than degree 3.

Errors: DM_BAD_CONT_FLAG_VALUE
The cont_flag value must be 0 or 1.

DM_NOT_BSPLINE_OR_NURB
The deformable model must be a B–spline or a NURB in order to modify output variables.

DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_eval_crv_src_domain

Function: Deformable Modeling

Action: Evaluate an embedded source curve for the given tag.

Prototype:
```
void DM_eval_crv_src_domain (
    int& rtn_err,           // error code
    DS_dmod* dmod,          // tag object ID
    int tag,                // tag object ID
    const double* s,        // array of domain points
    int npts,               // number of eval points
    DM_dbl_array& C,        // array of uv points
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dm_dbl_array.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: This function evaluates an embedded source curve. It returns the embedding surface *uv* (domain) points corresponding to the input curve parameter points.See DM_eval_dmod for evaluating the embedding surface. rtn_err will return 0 for success, or a negative error code. The two arrays are sized: domain points, sized:[npts]; uv points, sized:[C.Size()].

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_TAG_OBJECT_NOT_FOUND
Could not find tag object with the given tag.

DM_TAGOBJ_NO_SRC_PFUNC_OR_FUNC
Could not find src_C_pfunc or src_CW_func for given tag object (see DM_add_crv_cstrn, DM_add_crv_load, DM_add_link_cstrn).

DM_TAGOBJ_NOT_A_CURVE
Given tag object is not a curve.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_eval_crv_tgt

Function: Deformable Modeling

Action: Evaluate a target curve for the given tag. Returns the *xyz* points corresponding to the input curve parameter points.

Prototype:
```
void DM_eval_crv_tgt (
    int& rtn_err,           // error code
    DS_dmod* dmod,          // tag object ID
    int tag,                // tag object ID
    const double* s,        // array of domain points
    int npts,               // number of eval points
    DM_dbl_array& W,        // array of uv points
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dm_dbl_array.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: This function evaluates a target curve for the given tag.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_TAG_OBJECT_NOT_FOUND
Could not find tag object with the given tag.

DM_TAGOBJ_NO_SRC_PFUNC_OR_FUNC
Could not find src_W_pfunc, src_CW_func, or src_C_pfunc for given tag object (see DM_add_crv_cstrn, DM_add_crv_load, DM_add_link_cstrn).

DM_TAGOBJ_NOT_A_CURVE
Given tag object is not a curve.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_eval_dmod

Function: Deformable Modeling, Deformable Model Management

Action: Calculates image positions for the *uv* location and returns 0 or an error.

Prototype:

```
void DM_eval_dmod (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: dmod to evaluate
    int domain_flag,        // in: define dpt space
                            // 0=orig_dmod_space,
                            // 1=unit_space, 2=
                            // internal_pfunc_space.
    double* dpt,            // in: domain point for
                            // evaluation
                            // sized:[domain_dim]
    double* W,              // out: dmod xyz position
                            // for dpt loc
                            // W=[x,y,z],
                            // sized:[image_dim]
    double* dWu             // out: u_dir tang or
        = NULL,             // NULL
                            // dWu=[dx,dy,dz],
```

```
                                    // sized:[image_dim]
    double* dWv                     // out: v_dir tang or
        = NULL,                     // NULL
                                    // dWv=[dx,dy,dz],
                                    // sized:[image_dim]
    double* dWuu                    // out: uu parameter
        = NULL,                     // derivative or NULL
                                    // dWuu=[dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuv                    // out: uv parameter
        = NULL,                     // derivative or NULL
                                    // dWuv=[dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWvv                    // out: uu parameter
        = NULL,                     // derivative or NULL
                                    // dWvv=[dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuuu                   // out: uuu parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuuv                   // out: uuv parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuvv                   // out: uvv parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWvvv                   // out: vvv parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    SDM_options* sdmo               // in:SDM_options pointer
        = NULL                      // note: sets active dmod
                                    // note: any output array
                                    // set to NULL is not
                                    // computed or stored
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”dshusk/dskernel/dmapi.hxx”
#include ”dshusk/dskernel/dsdmod.hxx”
#include ”dshusk/dskernel/sdm_options.hxx”
```

Description: Modifies W, dWu, dWv, dWuu, dWuv, dWvv, dWuuu, dWuuv, dWuvv, dWvvv.

This function is a convenience routine for accessing DM_eval_pfunc given a dmod. This function extracts the input dmod's DS_pfunc pointer and passed the call along to DM_eval_pfunc.

The DM_eval_dmod function uses the dmod's contained pfunc to evaluate the image space values for the input domain point location.

When domain_flag == 0, dpt is given in the dmod's original domain range. When domain_flag == 1, dpt is given in the unit–square range. When domain_flag == 2, dpt is given in the pfunc's internal domain range.

Note, the orig_dmod_space is invariant, while the internal_pfunc_space may vary. See DM_get_dmod_domain_scale().

The function can compute and store the following,

W	Position [x,y,z]
dWu	1st parametric derivative in *u* direction
dWv	1st parametric derivative in *v* direction
dWuu	2nd parametric derivative in *uu* direction
dWuv	2nd parametric derivative in *uv* direction
dWvv	2nd parametric derivative in *vv* direction
dWuuu	3rd parametric derivative in *uuu* direction
dWuuv	3rd parametric derivative in *uuv* direction
dWuvv	3rd parametric derivative in *uvv* direction
dWvvv	3rd parametric derivative in *vvv* direction

The values for any output array set to NULL are not computed or stored. All output arrays, except for *W*, the position, default to NULL and may be omitted.

Surfaces can load all six output variables, Curves will only load W, dWu, and dWuu and will skip the others.

Errors: DM_NULL_INPUT_PTR
The dpt input pointer cannot be NULL.

DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_NULL_OUTPUT_PTR
The alpha cannot be NULL on entry.

DM_NON_NULL_OUTPUT_PTR
The output pointer cannot be NULL on entry.

DM_BAD_DOMAIN_DIM
The domain dimension must equal 1 or 2.

DM_parse_tag_flag() errors

DM_scale_unit_dpt_to_pfunc() errors

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_eval_pfunc

Function: Deformable Modeling, Deformable Model Management

Action: Calculates image positions for the input domain location and returns 0 or an error.

Prototype:

```
void DM_eval_pfunc (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_pfunc* pfunc,        // in: pfunc to evaluate
    int domain_flag,        // in: 2=domain_pts in
                            // internal_pfunc_space,
                            // 1=domain_pts in
```

```
                                    // unit_space.
    double* dpt,                    // in: domain point for
                                    // evaluation
                                    // sized:[domain_dim]
    double* W,                      // out: dmod xyz position
                                    // for dpt loc
                                    // W=[x,y,z],
                                    // sized:[image_dim]
    double* dWu                     // out: u_dir tang or
        = NULL,                     // NULL, dWu=[dx,dy,dz],
                                    // sized:[image_dim]
    double* dWv                     // out: v_dir tang or
        = NULL,                     // NULL, dWv=[dx,dy,dz],
                                    // sized:[image_dim]
    double* dWuu                    // out: uu parameter
        = NULL,                     // derivative or NULL
                                    // dWuu=[dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuv                    // out: uv parameter
        = NULL,                     // derivative or NULL
                                    // dWuv=[dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWvv                    // out: uu parameter
        = NULL,                     // derivative or NULL
                                    // dWvv=[dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuuu                   // out: uuu parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuuv                   // out: uuv parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWuvv                   // out: uvv parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    double* dWvvv                   // out: vvv parametric
        = NULL,                     // derivative or NULL
                                    // [dxx,dyy,dzz],
                                    // sized:[image_dim]
    SDM_options* sdmo               // in:SDM_options pointer
        = NULL                      // note: sets active dmod
                                    // note: any output array
```

```
                                    // set to NULL is not
                                    // computed or stored
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dspfunc.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Modifies W, dWu, dWv, dWuu, dWuv, dWvv, dWuuu, dWuuv, dWuvv, dWvvv.

This function uses the input pfunc to evaluate the image space values for the input domain point location. When domain_flag == 2, the input domain point is expected to be in the current domain range of the input pfunc. When domain_flag == 1, the the input domain point is assumed to be given in the unit–square range and the domain point is scaled to the actual pfunc domain range prior to being used in the evaluator. The function can compute and store the following,

W	Position [x,y,z]
dWu	1st parametric derivative in *u* direction
dWv	1st parametric derivative in *v* direction
dWuu	2nd parametric derivative in *uu* direction
dWuv	2nd parametric derivative in *uv* direction
dWvv	2nd parametric derivative in *vv* direction
dWuuu	3rd parametric derivative in *uuu* direction
dWuuv	3rd parametric derivative in *uuv* direction
dWuvv	3rd parametric derivative in *uvv* direction
dWvvv	3rd parametric derivative in *vvv* direction

The values for any output array set to NULL are not computed or stored. All output arrays, except for W, the position, default to NULL and may be omitted.

Surfaces can load all six output variables. Curves will only load W, dWu, and dWuu and will skip the others.

Errors: DM_NULL_INPUT_PTR
The dpt input pointer cannot be NULL.

DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_NULL_OUTPUT_PTR
The alpha cannot be NULL on entry.

DM_NON_NULL_OUTPUT_PTR
W output pointer cannot be NULL on entry.

DM_BAD_DOMAIN_DIM
The domain dimension must equal 1 or 2.

DM_parse_tag_flag()
errors

DM_scale_unit_dpt_to_pfunc()
errors

DM_BAD_DOMAIN_PT_RANGE
The domain point must be completely contained by the deformable model.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_extrapolate_dmod

Function: Deformable Modeling, Deformable Model Management

Action: Extends the deformable model domain by a small amount and returns 0 for success or an error.

Prototype:

```
void DM_extrapolate_dmod (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // i/o: target dmod
                            // to modify
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Modifies deformable model and possibly deformable model–>Parent().

Extends the domain of the underlying shape representation by 5% in all directions for deformable models. If the original domain of a curve was 100 <= *u* <= 200 then the modified domain would be 95 <= *u* <= 205. If the deformable model is a child patch and the extension in its domain would make it exceed the domain of its parent, then the parent's domain is also extended.

The shape of the deformable model in the extended region is defined by the edge element's original shape function. For B–splines and NURBs these shape functions are usually poorly behaved in regions far from the original domain. So repeated use of this function may build some ill–behaved looking shapes including self–intersections. However, use of the deformable modeling approach to reshape the extended shape with appropriate loads and constraints should make it look nice and smooth again. (i.e., run DM_solve())

A deformable model within a multi–surface mesh will not be extrapolated. Trying to do so will not change the database in any manner and may result in setting rtn_err = DM_MULTI_SURF_EXTRAPOLATE.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_MULTI_SURF_EXTRAPOLATE
When DM_extrapolate_dmod is called on a deformable model that is already within a multi–surface mesh.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Changes model

DM_find_cstrn_by_tag

Function: Deformable Modeling, DML Constraints, DML Tags

Action: Checks the entire hierarchy for a constraint with matching tag value.

Prototype:

```
DS_cstrn* DM_find_cstrn_by_tag (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: member of dmod
                            // hierarchy to search
    int tag,                // in: unique tag
                            // identifier to match
    int& patch_tag1,        // out: containing patch
                            // tag or err code
    int& patch_tag2,        // out: containing patch
                            // tag or err code
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              // note: sets active dmod
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dscstrn.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Searches every constraint in every deformable model in the hierarchy for a constraint with a matching tag. If found, this function returns the pointer to the constraint and updates the active deformable model pointer value. If the constraint is not a link constraint, patch_tag1 is set with the tag identifier of the deformable model containing the identified constraint and patch_tag2 is set to –1. When the constraint is a link constraint, patch_tag1 and patch_tag2 are set to the two deformable models linked by the constraint.

When a constraint with a matching tag identifier is not found, patch_tag1 and patch_tag2 are both set to –1 and NULL is returned.

Errors: DM_NULL_INPUT_PTR
The deformable model cannot be NULL on entry.

DM_TAG_OBJECT_NOT_FOUND
The queried deformable modeling hierarchy does not contain a deformable model, a load, or a constraint with a tag value that matches the input tag value.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_find_load_by_tag

Function: Deformable Modeling, DML Loads, DML Tags

Action: Checks the entire hierarchy for a load with matching tag value.

Prototype:
```
DS_load* DM_find_load_by_tag (
    int& rtn_err,           // out: 0=success or neg
                            // err code
    DS_dmod* dmod,          // in: member of dmod
                            // hierarchy to search
    int tag,                // in: unique tag
                            // identifier to match
    int& patch_tag,         // out: containing patch
                            // tag or err code
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              // note: sets active dmod
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”dshusk/dskernel/dmapi.hxx”
#include ”dshusk/dskernel/dsdmod.hxx”
#include ”dshusk/dskernel/dsload.hxx”
#include ”dshusk/dskernel/sdm_options.hxx”
```

Description: Searches every load in every deformable model in the hierarchy for a load with a matching tag. If found, this function returns the pointer to the load and updates the active deformable model pointer value.

The patch tag is set to a positive tag value for success.

Errors: DM_NULL_INPUT_PTR
The deformable model or alpha cannot be NULL on input.

DM_TAG_OBJECT_NOT_FOUND
The input tag cannot identify a deformable model, a load, a spring, a spring set, an attractor object, or a constraint in the model hierarchy.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_find_patch_by_entity

Function: Deformable Modeling, DML Patches

Action: Finds and returns pointer to DS_dmod deformable model whose entity pointer matches the input entity pointer or returns NULL.

Prototype:
```
DS_dmod* DM_find_patch_by_entity (
    int& rtn_err,           // out: 0=success
                            // or negative err code
    DS_dmod* dmod,          // in: member of
                            // hierarchy to search
    void* entity,           // in: tgt_dmod entity
                            // value to match
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Searches the deformable model's hierarchy for the deformable model whose entity pointer value is the same as the input entity pointer value. When found, the pointer to that deformable model is returned. When not found NULL is returned. The entire deformable modeling hierarchy containing the input dmod is automatically searched.

Errors: DM_NULL_INPUT_PTR
when dmod is NULL on entry.

DM_TAG_OBJECT_NOT_FOUND
No DS_dmod deformable model with a matching entity pointer was found.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_find_patch_by_tag

Function: Deformable Modeling, DML Patches, DML Tags

Action: Finds and returns the pointer to the DS_dmod deformable model whose tag matches the input tag value or returns NULL.

Prototype:
```
DS_dmod* DM_find_patch_by_tag (
    int& rtn_err,           // out: 0=success
                            // or negative err code
    DS_dmod* dmod,          // in: member of
                            // hierarchy to search
    int tag,                // in: the tag value
                            // to find
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Searches the deformable model's hierarchy for the deformable model whose tag value is the same as the input tag value. When found, the pointer to that deformable model is returned and that deformable model is made the active deformable model. When not found NULL is returned. The entire hierarchy containing the input dmod is automatically searched.

Errors: DM_NULL_INPUT_PTR
When dmod is NULL on input.

DM_TAG_NOT_A_DMOD_MEMBER
The tag_flag does not identify a deformable model in the patch hierarchy.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only

DM_find_patch_by_tag_flag

Function: Deformable Modeling, DML Patches, DML Tags

Action: Converts a tag_flag identifier into a deformable model pointer and a walk_flag value.

Prototype:

```
DS_dmod* DM_find_patch_by_tag_flag (
    int& rtn_err,           // out: 0=success
                            // or negative err code
    DS_dmod* dmod,          // in: member of
                            // hierarchy to search
    int tag_flag,           // in: specify target
                            // dmod and how deep to
                            // go. 1=active, 2=root,
                            // else = member
                            // (tag=tag_flag) pos=tgt
                            // only, neg=tgt and
                            // offspring
    int& walk_flag,         // out: 0=tgt only
                            // 1=tgt and offspring
                            // 2=tgt, siblings, and
                            // offspring.
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              // note: makes found
                            // patch active
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "dshusk/dskernel/dmapi.hxx"
#include "dshusk/dskernel/dsdmod.hxx"
#include "dshusk/dskernel/sdm_options.hxx"
```

Description: Modifies walk_flag, returns pointer to identified deformable model.

tag_flag identifies the target deformable model as follows. When tag_flag is equal to +1 or –1, the target is the active deformable model. When tag_flag is equal to +2 or –2, the target is the root deformable model. When tag_flag is equal to some other value, then the target is the deformable model that has the same tag identifier as the absolute value of the tag_flag value. When there is no deformable model whose tag identifier matches the absolute value of the tag_flag value, then there is no target and NULL is returned.

In summary:

+1 or –1	active deformable model
+2 or –2	root deformable model
+ or –num	deformable model with tag_id == |num|

The sign of the tag_flag value specifies the value of the walk_flag. walk_flag is a typical input required by all functions which are capable of walking the deformable modeling hierarchy recursively.

Valid walk_flag values include:

0 = operate only on the target deformable model.
1 = operate only on the target deformable model and its offspring.
2 = operate on the target deformable model, its younger siblings, and all their offspring.

For multisurface models: When input tag_flag value is –2 (root dmod and offspring), then the walk_flag value is set to 2 (target, siblings, and offspring), and the returned deformable model pointer is set to the oldest root sibling. When input tag_flag value is –1, the returned deformable model pointer is set to the currently active deformable model and walk_flag is set to 2 if the active Deformable model is a root sibling, else walk_flag is set to 1. When the input tag_flag value is any negative number other than –1 or –2, walk_flag is set to 1. When the input tag_flag value is positive, walk_flag is set to 0.

Errors: Returns NULL for any errors, and sets rtn_err to one of the following possible error values:

DM_DMOD_NOT_A_ROOT_DMOD
The deformable model must be the root of a hierarchy tree.

DM_BAD_TAG_FLAG_VALUE
The tag_flag cannot equal 0.

DM_NO_ACTIVE_DMOD
The input deformable model's hierarchy has no currently active deformable model.

DM_TAG_NOT_A_DMOD_MEMBER
The tag_flag does not identify a deformable model in the patch hierarchy.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: System routine

DM_find_tag_by_image_line

Function: Deformable Modeling, DML Tags, DML Graphics

Action: Checks the entire hierarchy for the load, constraint, or control point closest to the input image_line.

Prototype:

```
int DM_find_tag_by_image_line (
    int& rtn_err,           // out: 0=success
                            // or negative err code
    DS_dmod* dmod,          // in: member of
                            // hierarchy to search
    double* p0,             // in: p0 of iline =
                            // p0 + u*(p1–p0)
                            // sized:[image_dim]
    double* p1,             // in: p1 of iline =
                            // p0 + u*(p1–p0)
                            // sized:[image_dim]
    double max_dist,        // in: width of ray
                            // in which to search
    int& pt_index,          // out: image_pt index
                            // in tag for closest
                            // point
    SDM_options* sdmo       // in:SDM_options pointer
        = NULL              //
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”dshusk/dskernel/dmapi.hxx”
#include ”dshusk/dskernel/dsdmod.hxx”
#include ”dshusk/dskernel/sdm_options.hxx”
```

Description:

This function finds and returns the closest load, constraint, or control point tag object to an image space line.

The intent of this function is to enable mouse–based selection of tag objects. This works well when the image line (p0 and p1) is defined by the mouse position and the viewing angle. Select p0 and p1 to be two points on the line which intersects the mouse cursor position and runs in the viewing direction. Make sure that p0 is closer to the view than p1. The object selected by this routine will appear to be the one pointed to by the mouse.

Searches all the tag objects (pressure, constraints, control points, etc.) that are currently displayed (as recorded by deformable model–>Draw_state()). The entire patch hierarchy is searched. The width of the search ray is given in image space units. Applications with access to window information will want to compute this distance from pixel size information. Usually five pixels is a good ray thickness. Only those tag objects within the search ray thickness are considered for selection.

Returns the tag of the tag object closest to the viewer (lowest image line *u* parameter value) within the bounded search volume. Loads and constraints have tag identifiers greater than 0, and control points have tag identifiers less than or equal to –500.

Errors: DM_TAG_OBJECT_NOT_FOUND
The input tag cannot identify a deformable model, a load, a spring, a spring set, an attractor object, or a constraint in the model hierarchy.

DM_NO_ROOT_DMOD
The root deformable model cannot be NULL on input.

DM_NULL_INPUT_PTR
The value of P0 or p1 cannot be NULL on entry

DM_BAD_MAX_DIST_VALUE
The max_dist must be larger than 0.0.

Limitations: None

Library: dshusk

Filename: ds/dshusk/dskernel/dmapi.hxx

Effect: Read–only