UF_BREP_ask_edge_class (view source)

Defined in: uf_brep.h

Overview

This function returns the class numbers for the edge.

A ring edge is an edge, like the end of a cylinder, that has a curve but no vertices. A closed edge is like a ring edge but has one vertex. An open edge has two vertices. In topology structures, it is permissible for a closed edge to have two vertices as children but they must be the same vertex.

Environment

Internal and External

See Also

UF_BREP_ask_geometry
Refer to table

Required License(s)

gateway

```
int UF_BREP_ask_edge_class
(
    const UF_BREP_topology_t * item,
    int * edge_class,
    int * geom_class
)
```

const UF_BREP_topology_t *	**item**	Input	Item of topology for which the edge class numbers are to be returned.
int *	**edge_class**	Output	Class number of edge: UF_BREP_EDGE_OPEN UF_BREP_EDGE_CLOSED UF_BREP_EDGE_RING
int *	**geom_class**	Output	Class number of geometry on edge: UF_BREP_EDGE_NORMAL: has 3D curve. UF_BREP_EDGE_INTERSECTION: has 3D curve derived from surfaces. UF_BREP_EDGE_TOLERANT: has SP curve. UF_BREP_EDGE_SP: not a tolerant edge but represented with an SP curve.

UF_BREP_ask_geometry (view source)

Defined in: uf_brep.h

Overview

The given geometry is queried from the model and returned in the application-supplied structure. Any arrays needed (for example in b-surfaces and extruded surfaces) are allocated by Open API and must be freed with UF_free, or with UF_BREP_free_geometry_data.

For curves, geom_sense is forward if the curve tangent is pointing in the same direction as traversing the edge from the start vertex to the end vertex. For surfaces, geom_sense is forward if the surface normal points in the face normal direction (which the loop direction determines).

Special attention is given to geometries that cannot be returned without approximation. Blend surfaces, intersection edges, and tolerant edges have no direct representation in available geometry types. Therefore, when requesting data for a face or edge, the mappings and tolerances must always be supplied in the respective structures. Applications sometimes provide different tolerances for faces and edges, or for different types of surfaces or curves.
This function's results are undefined if any function that changes the body's topology is used between the time UF_BREP_ask_topology is called and the time this function is called. This includes any feature changes or any other modelling operation.

Geometry may be asked on a topology being used to create a body. It is important to realize that when asking for fin geometry for a tolerant edge, the SP curves returned were produced at the time the edge was made tolerant (when it was modelled). Its accuracy was determined at that time. Such curves are returned by this function as they were constructed when made. Setting a smaller tolerance for this function does not improve their accuracy. A smaller tolerance in this function only improves the accuracy of curves or surfaces that must be approximated by this function.

Environment
Internal and External

See Also
UF_BREP_ask_topology

Required License(s)
gateway

```
int UF_BREP_ask_geometry
(
    const UF_BREP_topology_t * topology,
    const UF_BREP_options_t * options,
    const UF_BREP_mapping_t * mapping,
    UF_BREP_geometry_t * geometry,
    UF_BREP_orientation_t * geom_sense,
    int * num_states,
    UF_BREP_state_t * * states
)
```

const UF_BREP_topology_t *	**topology**	Input	Pointer to topology item structure defining geometry to be returned
const UF_BREP_options_t *	**options**	Input	Applicable options: All tolerances for edge and fin curves, and for surfaces, should be specified.
const UF_BREP_mapping_t *	**mapping**	Input	Surface and curve type mappings to cause output to be in b-spline geometry.
UF_BREP_geometry_t *	**geometry**	Input	Pointer to geometry structure to receive the description.

UF_BREP_orientation_t *	**geom_sense**	Input / Output	Input: Geometry orientation with respect to the topology entity. For faces, fins, and edges: UF_BREP_ORIENTATION_FORWARD or UF_BREP_ORIENTATION_REVERSE. Otherwise UF_BREP_ORIENTATION_NONE. Output: Pointer to geometry structure to receive the description.
int *	**num_states**	Output	number of states in the states array
UF_BREP_state_t * *	**states**	Output to UF_*free*	UF_BREP allocated array. If the caller passes in a NULL, UF_BREP does not allocate the array. This should be freed by calling UF_free.

UF_BREP_ask_identifier (view source)

Defined in: uf_brep.h

Overview

Returns a unique identifier within a body for topology elements that do not have NX tags, viz., vertices, fins, and shells. The identifier is useful for Open API level algorithms that need to test for identical topology elements, e.g., is a vertex on an edge the same as a vertex on another edge?

Return

void

Environment

Internal and External

Required License(s)

gateway

```
void UF_BREP_ask_identifier
(
    const UF_BREP_topology_t * body_topos,
    int * identifier
)
```

const UF_BREP_topology_t *	**body_topos**	Input	Topology entity for which to return an identifier.
int *	**identifier**	Output	Unique for all other topology identifiers within the same body.

UF_BREP_ask_topology (view source)

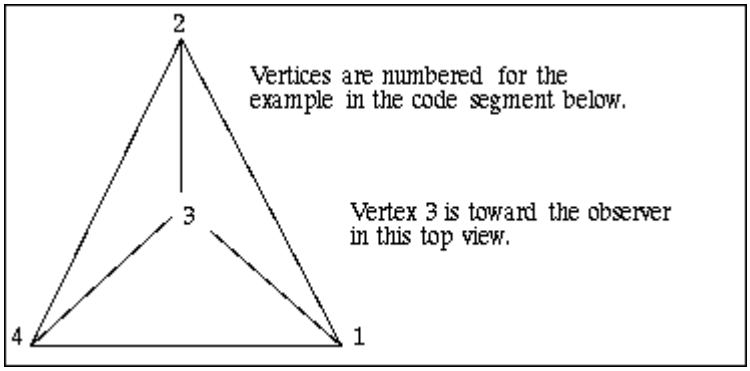
Defined in: uf_brep.h

Overview

Returns a topological description of a body or face. The body_topos list with the children pointer arrays defines the connections between the entities of the body.

The topology for a face is returned as if it were a sheet body, unless SP curve modifications are requested, in which case the vertices are omitted.

The orientation of edges or fins in loops is indicated by the orientation member of the oriented child structure.



Environment

Internal and External

See Also

See UF_BREP_ask_geometry
UF_BREP_release_topology
UF_BREP_ask_edge_class
The example is a tetrahedron, for which the topology is read out. See ufx_brep.c for a more complex example that calls more UF_BREP functions to create a copy of a solid body and prints out the indented topology tree.

Required License(s)

gateway

```
int UF_BREP_ask_topology
(
    tag_t body,
    const UF_BREP_options_t * options,
    UF_BREP_topology_t * * body_topos,
    int * num_states,
    UF_BREP_state_t * * states
)
```

tag_t	**body**	Input	Tag of body for which to return topology

const UF_BREP_options_t *	**options**	Input	Options are specified as needed. Fin modifications (flattening, singularities) are specified here when requesting surface data for trimming.
UF_BREP_topology_t * *	**body_topos**	Output to UF_*free*	Topology structures and arrays that describe the body.
int *	**num_states**	Output	Number of states in the state array
UF_BREP_state_t * *	**states**	Output to UF_*free*	UF_BREP allocated array. If the caller passes in a NULL, UF_BREP does not allocate the array. This should be freed by calling UF_free.

UF_BREP_ask_topology_source (view source)

Defined in: uf_brep.h

Overview
Returns the source of the specified topology.

Environment
Internal and External

See Also
UF_BREP_validate_topology

Required License(s)
gateway

```
int UF_BREP_ask_topology_source
(
    const UF_BREP_topology_t * body_topos,
    UF_BREP_topo_source_t * topo_source
)
```

const UF_BREP_topology_t *	**body_topos**	Input	Address of topology array whose state is to be returned
UF_BREP_topo_source_t *	**topo_source**	Output	Source of topology

---

UF_BREP_attach_geometry (view source)

Defined in: uf_brep.h

Overview
The given geometry is created and attached to the topology item indicated. UF_BREP_attach_geometry returns an error if the geometry

already has attached geometry.

No geometry is permitted to be attached to a topology item not of the correct type. Only edges and fins may have curves, only faces may have surfaces, and only vertices may have points. You may attach 3D curves to edges and SP curves to the fins of that edge. The additional information is sometimes useful in creating a valid edge.
For curves, geom_sense is forward if the curve tangent is pointing in the same direction as traversing the edge from the start vertex to the end vertex. For surfaces, geom_sense is forward if the surface normal points in the face normal direction (which the loop direction determines).

Simplifications performed on b-geometry are the following:
A bspline that is linear is converted to a line. This is done for linear and quadratic spline data.

A bspline that represents a circle is converted to a circle.

A surface of revolution with a line as the generator is converted to a cylinder or cone, if the generator is aligned suitably.

A surface of revolution with an arc as the generator is converted to a sphere or torus, if the generator is aligned suitably. Even partial arcs are simplified to the complete surface.

A bsurface that is planar is converted to a plane. This is done for linear and quadratic bsurface data.

A bsurface that is linear in one direction and circular in the other is converted to a cylinder or cone.

A bsurface that is circular in both directions is converted to a sphere or torus.

Please note the following items:

Some configurations of spline data that represent analytic geometry may not be simplified. This is because particular forms may not be recognized by Open API.

If simplification is requested, and SP curves are to be used to construct the face, it is necessary that the simplification introduce a transformation of the surface parameterization that is linear in nature. Any other transformation causes SP curves to assume an incorrect shape. In particular, conversions of b-surfaces and b-curves to their analytic equivalents may introduce parameterization changes that are incompatible with using SP curves for face construction. If the latter is the case, a state that so indicates is returned.

Environment

Internal and External

See Also

UF_BREP_validate_topology

Required License(s)

solid_modeling

```
int UF_BREP_attach_geometry
(
    UF_BREP_topology_t * topology,
    const UF_BREP_geometry_t * geometry,
    const UF_BREP_orientation_t geom_sense,
    const UF_BREP_options_t * options,
    int * num_states,
    UF_BREP_state_t * * states
)
```

UF_BREP_topology_t *	**topology**	Input	Pointer to topological structure containing item to which the geometry is to be attached
const UF_BREP_geometry_t *	**geometry**	Input	Pointer to geometry structure defining geometry to be attached
const UF_BREP_orientation_t	**geom_sense**	Input	Geometry orientation with respect to the topology entity. For faces, fins, and edges use: UF_BREP_ORIENTATION_FORWARD or UF_BREP_ORIENTATION_REVERSE. Otherwise use UF_BREP_ORIENTATION_NONE.
const UF_BREP_options_t *	**options**	Input	Applicable options: Simplification, UF_BREP_simplify_option
int *	**num_states**	Output	number of states in the states array
UF_BREP_state_t * *	**states**	Output to UF_*free*	UF_BREP allocated array. If the caller passes in a NULL, UF_BREP does not allocate the array. This should be freed by calling UF_free.

UF_BREP_delete_geometry (view source)

Defined in: uf_brep.h

Overview

Deletes the geometry attached to the specified topology. Note that the delete is permissible only on topologies that have not yet become a NX body. Typically a UF_BREP_attach_geometry call would follow a UF_BREP_delete_geometry call, e.g., if the attach failed for a known and correctable reason.

Environment

Internal and External

See Also

UF_BREP_attach_geometry
UF_BREP_release_topology

Required License(s)

solid_modeling

```
int UF_BREP_delete_geometry
(
    const UF_BREP_topology_t * item
)
```

const UF_BREP_topology_t *	**item**	Input	Item for which geometry is to be deleted.

UF_BREP_free_geometry_data (view source)

Defined in: uf_brep.h

Overview
All data structures allocated by Open API when UF_BREP_ask_geometry was called are freed.

Environment
Internal and External

See Also
UF_BREP_ask_geometry

Required License(s)
solid_modeling

```
int UF_BREP_free_geometry_data
(
    UF_BREP_geometry_p_t geometry,
    UF_BREP_free_fn_t fn
)
```

UF_BREP_geometry_p_t	**geometry**	Input / Output	Data structures allocated by UF_BREP_ask_geometry are freed.
UF_BREP_free_fn_t	**fn**	Input	UF_BREP_free_geometry_data calls UF_BREP_free_fn_t to free geometry structure data, e.g., B-curve and B-surface pole arrays. If free_function is NULL, then this routine uses UF_free.

UF_BREP_heal_body (view source)

Defined in: uf_brep.h

Overview
This routine fixes, or cleans up, a body created by other Open API functions. It attempts to re-intersect edges, reposition vertices, and set tolerances as designated by the level specified, to make the body pass all checks. Any object (typically a face or edge) that cannot be fixed is returned in the state list.

The tolerance value specified is an upper limit. Any entity that

requires a tolerance larger than this limit in order to pass checks is returned in the states array.

Any given object may have more than one entry in the states array. The healing levels cause the following fix ups to be attempted:
0 Report problems with the body in the state list.
1 For all problems fixable with tolerances, apply the smallest tolerance that alleviates the problem.
2 For any edge that must be tolerant at level 1, use surface-surface intersection to attempt to create an exact curve for the edge. Any that cannot be made exact are made tolerant.
3 For any edge that must be tolerant at level 1 and for all spline edges, use surface-surface intersection to attempt to create an exact curve for the edge. Any that cannot be made exact are made tolerant.
4 For all edges, use surface-surface intersection to attempt to create an exact curve for the edge. Any that cannot be made exact are made tolerant.

Environment
Internal and External

Required License(s)
solid_modeling

```
int UF_BREP_heal_body
(
    tag_t body,
    int level,
    double toler,
    int * num_states,
    UF_BREP_state_t * * states
)
```

tag_t	**body**	Input	body to fix
int	**level**	Input	healing level: any of 0 through 4
double	**toler**	Input	maximum tolerance; fix up of entities requiring tolerances exceeding this are returned in the state list
int *	**num_states**	Output	number of states in the states array
UF_BREP_state_t * *	**states**	Output to UF_*free*	UF_BREP allocated array. If the caller passes in a NULL, UF_BREP does not allocate the array. This should be freed by calling UF_free.

UF_BREP_make_body (view source)

Defined in: uf_brep.h

Overview
Creates an NX body from a UF_BREP topology hierarchy that defines the body topology and geometry. In addition to returning the NX body tag,

UF_BREP_make_body assigns the NX edge and face tags to the corresponding input UF_BREP topology structures. UF_BREP_make_body also does the equivalent of UF_BREP_heal_body to help make the NX body pass validity checks.

With solid topology input, the output is an NX solid body. With face topology input, the output is an NX sheet body. For a multi-face sheet topology, you must call UF_BREP_make_body for each face of the sheet, and then use UF_MODL_create_sew to sew the sheets together. You can call UF_BREP_make_body with a sheet topology, but the resulting NX sheet body represents only the first face of the input sheet topology.

Environment
Internal and External

See Also
UF_BREP_heal_body
UF_BREP_validate_topology

Required License(s)
solid_modeling

```
int UF_BREP_make_body
(
    UF_BREP_topology_t * topology,
    tag_t * body,
    int * num_states,
    UF_BREP_state_t * * states
)
```

UF_BREP_topology_t *	**topology**	Input	Topology that has all geometry attached. The toplogy is made into an NX body. You can pass a face topology element.
tag_t *	**body**	Output	NX tag of body
int *	**num_states**	Output	number of states in state array
UF_BREP_state_t * *	**states**	Output to UF_*free*	UF_BREP allocated array. If the caller passes in a NULL, UF_BREP does not allocate the array. This should be freed by calling UF_free.

UF_BREP_release_topology (view source)

Defined in: uf_brep.h

Overview
Frees the UF_BREP structures of a topology. The structures include the oriented children structure arrays.

If a is topology built by the user and not finished, UF_BREP_release_topology must be called.

Environment

Internal and External

See Also
UF_BREP_validate_topology
UF_BREP_ask_topology

Required License(s)
solid_modeling

int UF_BREP_release_topology
(
const UF_BREP_topology_t * body_topos,
UF_BREP_free_fn_t fn
)

const UF_BREP_topology_t *	**body_topos**	Input	Topology for body the application is done with
UF_BREP_free_fn_t	**fn**	Input	UF_BREP_free_geometry_data calls UF_BREP_free_fn_t to free geometry structure data, e.g., B-curve and B-surface pole arrays. If the free_function is NULL, then this routine uses UF_free.

UF_BREP_validate_topology (view source)

Defined in: uf_brep.h

Overview
Optional function to validate the body_topos tree. It validates the children contents, i.e., shells must be the children of the body, faces must be children of shells, loops the children of faces, fins (if present) the children of loops, edges the children of fins (if present, children of loops if not), and vertices the children of edges (if edges are present) or fins (if edges are not present). Thus the type members must be set to valid values and the children arrays must be set (but the children arrays may be unspecified, when it is desirable to build the body in stages.)

The extension fields are updated with a pointer for use by the UF_BREP module. The application must not alter this value and must use it only with the macros designated for that purpose.

Environment
Internal and External

See Also
UF_BREP_attach_geometry

Required License(s)
solid_modeling

int UF_BREP_validate_topology
(
const UF_BREP_topology_t * body_topos,
int * num_states,

UF_BREP_state_t * * states
)

const UF_BREP_topology_t *	**body_topos**	Input	Topology structure array to initialize
int *	**num_states**	Output	Number of states in the state array
UF_BREP_state_t * *	**states**	Output to UF_*free*	UF_BREP allocated array. If the caller passes in a NULL, UF_BREP does not allocate the array. This should be freed by calling UF_free.