NAME

xgl_nurbs_surface – draws a NURBS surface

SYNOPSIS

#include <xgl/xgl.h>

```
void  xgl_nurbs_surface (
        Xgl_ctx                        ctx,
        Xgl_nurbs_surf                 *surface,
        Xgl_trim_loop_list             *trimming,
        Xgl_nurbs_surf_simple_geom     *hints, /* optional */
        Xgl_surf_color_spline          *color_spline, /* optional */
        Xgl_surf_data_spline_list      *data_splines ); /* not supported */
```

DESCRIPTION

This operator generates a NURBS surface of specified order based on the list of knots in the parameter space, list of control points, and the trimming information. Either rational or non-rational surface is selected, depending on the point type.

Parameters

ctx The XGL 3D Context containing the graphics state used when processing **xgl_-nurbs_surface**(3).

surface A pointer to an *Xgl_nurbs_surf* structure that is defined as follows:

```
typedef struct {
        Xgl_usgn32   order_u;         /* Order of surface in u */
        Xgl_usgn32   order_v;         /* Order of surface in v */
        Xgl_usgn32   num_knots_u;     /* Number of knots in u */
        float        *knot_vector_u;  /* Knot vector in u */
        Xgl_usgn32   num_knots_v;     /* Number of knots in v */
        float        *knot_vector_v;  /* Knot vector in v */
        Xgl_pt_list  ctrl_pts;        /* Row-ordered list of control
                                         points (index changes faster
                                         in u direction than v). */
} Xgl_nurbs_surf;
```

surface→order must be a positive integer.

surface→num_knots is the number of knots. In both U and V, the number of knots must be equal to the corresponding number of control points plus the order of the spline. For example, the number of knots in U is calculated as follows:

ctrl_pts_u + order_u

surface→knot_vector is an array of floats. In both U and V, the knot vector must form a non-decreasing sequence.

surface→ctrl_pts is an array of points. The weight of a rational control point cannot be less than or equal to zero. The number of control points is the product of the number of points in U and the number of points in V.

*(num_knots_u - order_u) * (num_knots_v - order_v)*

trimming

The trimming parameters define the trimmed portion of the displayed surface. If trimming is NULL, the whole surface between its U and V parametric ranges is displayed. Trimming information consists of a list of trim loops. Each trim loop consists of a list of trim curves, connected to each other in a head to tail fashion.

Xgl_trim_loop_list structure is defined as follows:

```
typedef struct {
        Xgl_usgn32       num_loops;
        Xgl_trim_loop    *trim_loops;
} Xgl_trim_loop_list;
```

list→num_loops is the number of trim loops.

list→trim_loops is a list of trim curves. It is a structure defined as:

```
typedef struct {
        Xgl_usgn32       num_curves;
        Xgl_trim_curve   *curves;
} Xgl_trim_loop;
```

loop→num_curves is the number of trim curves in a loop.

loop→curves is an array of 2D NURBS curves, defined in the parameter space of the surface.

Each trim loop must be closed, continuous and should not intersect itself or another trim loop. Moreover, valid trim loops must satisfy the odd winding rule and curve orientation rule. For more information about trimming, please refer to the *XGL Programmer's Guide*.

Xgl_trim_curve structure is defined as follows:

```
typedef struct {
        Xgl_usgn32              order;                    /* Order of curve */
        Xgl_usgn32              num_knots;                /* Number of
                                                             knots */
        float                   *knot_vector;             /* Knot vector */
        Xgl_bounds_f1d          range;                    /* Range of
                                                             parameter */
        Xgl_pt_list             ctrl_pts;                 /* List of control
                                                             points */
/*
 * Trim curve structure fields (applicable only to trim curves)
 */
        Xgl_boolean             trim_curve_vis;           /* Highlight trim
                                                             curve */
        Xgl_trim_curve_approx   trim_curve_approx;        /* Approx type */
        float                   trim_curve_approx_value;  /* Approx value */
} Xgl_trim_curve;
```

curve→order must be greater than or equal to 2.

curve→num_knots is the number of knots which must be equal to the corresponding number of control points plus the order of the trim curve.

curve→knot_vector is an array of floats. The knots must form a non-decreasing sequence.

curve→range is the range parameters defining the parametric limits of the trim curve.

Xgl_bounds_f1d structure is defined as follows:

```
typedef struct {
        float   bmin;
        float   bmax;
} Xgl_bounds_f1d;
```

The minimum value should not be greater than the maximum value, and both values should lie between *curve→knot_vector[order-1]* and *curve→knot_vector-[curve→num_knots-order]*.

curve→ctrl_pts is an array of points. Trim curve can be rational or non-rational, depending on the point type. The weight of a rational control point cannot be less than or equal to zero.

curve→trim_curve_vis controls the highlighting of the trim curve. If set, the trim curve will be highlighted using current edge attributes.

curve→trim_curve_approx and *trim_curve_approx_value* controls the quality of smoothness of a trim curve. The former selects the approximation method, and the latter selects the reference measure used in the approximation of the curve. If the Xgl_trim_curve_approx is XGL_TRIM_CURVE_CONST_PARAM_SUBDIV_BETWEEN_-KNOTS, trim curve is rendered with specified (by trim_curve_approx_value) number of segments between each adjacent pair of knots. If the Xgl_trim_-curve_approx is XGL_TRIM_CURVE_VIEW_DEPENDENT, trim curve tessellation is controlled by the tessellation of the surface. In this case, a value of *curve→trim_-curve_approx_value* between 0 and 1 indicates the relative quality of the trim curves compared to surface quality. Higher values mean better relative quality.

Xgl_trim_curve_approx structure is defined as:

```
typedef enum {
        XGL_TRIM_CURVE_CONST_PARAM_SUBDIV_BETWEEN_KNOTS,   /* static
                                                             tessellation */
        XGL_TRIM_CURVE_VIEW_DEPENDENT                       /* dynamic
                                                             tessellation */
} Xgl_trim_curve_approx;
```

hints An application may optionally supply hints about the shape of the surface. If the surface is planar, spherical, cylindrical, or conical, XGL can optimize the rendering. *Xgl_nurbs_surf_simple_geom* structure is defined as:

```
typedef struct {
        Xgl_nurbs_surf_type         surf_type;
        Xgl_nurbs_surf_geom_desc    geom_desc;
} Xgl_nurbs_surf_simple_geom;
```

The *geom→surf_type* structure for different nurbs surfaces is as follows:

```
typedef enum {
        XGL_SURF_NURBS,             /* default, no optimization */
        XGL_SURF_PLANAR,            /* surface is planar */
        XGL_SURF_CYLINDRICAL,       /* surface is cylindrical */
        XGL_SURF_CONICAL,           /* surface is conical */
        XGL_SURF_SPHERICAL,         /* surface is spherical */
} Xgl_nurbs_surf_type;
```

The *geom→geom_desc* structure for nurbs surface geometry is as follows:

```
typedef union {
        Xgl_plane           planar;
        struct {
                Xgl_pt_f3d  axial_pt;
                Xgl_pt_f3d  axis_dir;
                float       radius;
                Xgl_boolean norm_flag;
        } cylindrical;
        struct {
                Xgl_pt_f3d  apex;
                Xgl_pt_f3d  axis_dir;
                float       cone_angle;  /* 0 = zero width cone (line)*/
                Xgl_boolean norm_flag;
        } conical;
        struct {
                Xgl_pt_f3d  center;
                float       radius;
                Xgl_boolean norm_flag;
        } spherical;
} Xgl_nurbs_surf_geom_desc;
```

A planar surface is defined by a point on it and the normal. A cylindrical surface is defined by an axis, a point on the axis, and a radius. A conical surface is defined by its apex, axis, and an angle between the axis and the side. Finally, a spherical surface is defined by its center and a radius. The norm_flag for cylindrical, conical, and spherical surfaces indicates whether normals to the surface point outwards (TRUE) or inwards (FALSE).

All the NURBS parameters to xgl_nurbs_surface operator must always be provided even though simple geometry hints is specified. It is the application's responsibility to ensure that provided hint does not conflict with the geometry defined by the NURBS parameters. If this is not the case, lighting may be incorrect.

color_spline

The color_spline is a nurbs surface which describes the color distribution over the primitive's geometry. It is a pointer to an *Xgl_surf_color_spline* structure that is defined as follows:

```
typedef struct {
        Xgl_usgn32              order_u;
        Xgl_usgn32              order_v;
        Xgl_usgn32              num_knots_u;
        float                   *knot_vector_u;
        Xgl_usgn32              num_knots_v;
        float                   *knot_vector_v;
        Xgl_color_homogeneous   *colors;
} Xgl_surf_color_spline;
```

The order, num_knots, and knot_vector elements should meet the same requirement as structure *Xgl_nurbs_surface* above. Color splines need not have the same order, rationality or knot sequence as the geometry spline.

spline→colors is an array of control points of the color spline. The number of control points is calculated in the same way as the number of control points (*surface→ctrl_pts*) discussed above.

The control points of the color spline are color coordinates. The color coordinates could be in 3D (non-rational) or 3D+w (rational). The values of the color spline control point coordinates may be outside the range normally associated with the color value. The vertex color for any point on the mapped geometry of the surface can be determined by evaluating the color spline at the corresponding parameter value.

Xgl_color_homogeneous union is defined as follows:

```
typedef union {
        Xgl_color_rgbw   rgbw;
} Xgl_color_homogeneous;
```

It is a union of all color types (except index) with w coordinate.

Xgl_color_rgbw structure is defined as follows:

```
typedef struct {
        float   r, g, b;
        float   w;
} Xgl_color_rgbw;
```

If the non-rational color is used, w should be set to 1. The color type has to be consistent with the device color type and should not be indexed.

Attributes The quality of smoothness of the surface can be controlled with the **XGL_CTX_NURBS_-SURF_APPROX**(3), **XGL_CTX_NURBS_SURF_APPROX_VAL_U**(3) and **XGL_CTX_NURBS_SURF_-APPROX_VAL_V**(3) attributes. The first selects the approximation method, and the remaining two selects the reference measures used in the approximation of the surface. The minimum and the maximum number of segments used to display the surface can be controlled with **XGL_CTX_MIN_TESSELLATION**(3) and **XGL_CTX_MAX_TESSELLATION**(3) attributes.

The trim loops constitute the edges of the B-spline. If the B-spline is untrimmed, the parametric boundaries constitute the edges. Edges are rendered using current edge attributes. Individual trim curves may be turned on or off by using the *curve→trim_curve_vis* flag.

The **XGL_3D_CTX_SURF_FRONT_FILL_STYLE**(3) and the **XGL_3D_CTX_SURF_BACK_FILL_STYLE**(3) attributes control the appearance of the interior surface. If it is XGL_SURF_FILL_SOLID, the interior is filled entirely. If it is XGL_SURF_FILL_HOLLOW, the surface is rendered as lines along the edges. These lines have the surface color and may be shaded. If it is XGL_SURF_-FILL_EMPTY, then the interior is not drawn.

Optionally, iso-parametric curves and level curves (currently not implemented) may be rendered on top of the surface. Curves are rendered using current line attributes. Also, edges of approximating facets may be displayed using current edge attributes. The **XGL_-CTX_NURBS_SURF_PARAM_STYLE**(3) attribute controls the appearance of the surface.

RETURN VALUE

Nothing is returned.

SEE ALSO

XGL_CTX_NURBS_SURF_APPROX(3)
XGL_CTX_NURBS_SURF_APPROX_VAL_U(3)
XGL_CTX_NURBS_SURF_APPROX_VAL_V(3)
XGL_CTX_NURBS_SURF_PARAM_STYLE(3)
XGL_CTX_NURBS_SURF_ISO_CURVE_PLACEMENT(3)
XGL_CTX_NURBS_SURF_ISO_CURVE_U_NUM(3)
XGL_CTX_NURBS_SURF_ISO_CURVE_V_NUM(3)
XGL_CTX_MAX_TESSELLATION(3)
XGL_CTX_MIN_TESSELLATION(3)
XGL_CTX_EDGE_AA_BLEND_EQ(3)
XGL_CTX_EDGE_AA_FILTER_SHAPE(3)
XGL_CTX_EDGE_AA_FILTER_WIDTH(3)
XGL_CTX_EDGE_ALT_COLOR(3)
XGL_CTX_EDGE_CAP(3)
XGL_CTX_EDGE_COLOR(3)
XGL_CTX_EDGE_JOIN(3)
XGL_CTX_EDGE_MITER_LIMIT(3)
XGL_CTX_EDGE_PATTERN(3)
XGL_CTX_EDGE_STYLE(3)

XGL_CTX_EDGE_WIDTH_SCALE_FACTOR(3)
XGL_CTX_LINE_AA_BLEND_EQ(3)
XGL_CTX_LINE_AA_FILTER_SHAPE(3)
XGL_CTX_LINE_AA_FILTER_WIDTH(3)
XGL_CTX_LINE_ALT_COLOR(3)
XGL_CTX_LINE_CAP(3)
XGL_CTX_LINE_COLOR(3)
XGL_CTX_LINE_COLOR_SELECTOR(3)
XGL_CTX_LINE_JOIN(3)
XGL_CTX_LINE_MITER_LIMIT(3)
XGL_CTX_LINE_PATTERN(3)
XGL_CTX_LINE_STYLE(3)
XGL_CTX_LINE_WIDTH_SCALE_FACTOR(3)
XGL_3D_CTX_LINE_COLOR_INTERP(3)
XGL_3D_CTX_SURF_BACK_AMBIENT(3)
XGL_3D_CTX_SURF_BACK_COLOR(3)
XGL_3D_CTX_SURF_BACK_COLOR_SELECTOR(3)
XGL_3D_CTX_SURF_BACK_DIFFUSE(3)
XGL_3D_CTX_SURF_BACK_DMAP(3)
XGL_3D_CTX_SURF_BACK_DMAP_NUM(3)
XGL_3D_CTX_SURF_BACK_DMAP_SWITCHES(3)
XGL_3D_CTX_SURF_BACK_FILL_STYLE(3)
XGL_3D_CTX_SURF_BACK_FPAT(3)
XGL_3D_CTX_SURF_BACK_FPAT_POSITION(3)
XGL_3D_CTX_SURF_BACK_ILLUMINATION(3)
XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT(3)
XGL_3D_CTX_SURF_BACK_SPECULAR(3)
XGL_3D_CTX_SURF_BACK_SPECULAR_COLOR(3)
XGL_3D_CTX_SURF_BACK_SPECULAR_POWER(3)
XGL_3D_CTX_SURF_BACK_TRANSP(3)
XGL_3D_CTX_SURF_DC_OFFSET(3)
XGL_3D_CTX_SURF_FACE_CULL(3)
XGL_3D_CTX_SURF_FACE_DISTINGUISH(3)
XGL_3D_CTX_SURF_FRONT_AMBIENT(3)
XGL_3D_CTX_SURF_FRONT_DIFFUSE(3)
XGL_3D_CTX_SURF_FRONT_DMAP(3)
XGL_3D_CTX_SURF_FRONT_DMAP_NUM(3)
XGL_3D_CTX_SURF_FRONT_DMAP_SWITCHES(3)
XGL_3D_CTX_SURF_FRONT_ILLUMINATION(3)
XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT(3)
XGL_3D_CTX_SURF_FRONT_SPECULAR(3)
XGL_3D_CTX_SURF_FRONT_SPECULAR_COLOR(3)
XGL_3D_CTX_SURF_FRONT_SPECULAR_POWER(3)
XGL_3D_CTX_SURF_FRONT_TRANSP(3)
XGL_3D_CTX_SURF_GEOM_NORMAL(3)

XGL_3D_CTX_SURF_NORMAL_FLIP(3)
XGL_3D_CTX_SURF_SILHOUETTE_EDGE_FLAG(3)
XGL_3D_CTX_SURF_TRANSP_BLEND_EQ(3)
XGL_3D_CTX_SURF_TRANSP_METHOD(3)
XGL_CTX_SURF_EDGE_FLAG(3)
XGL_CTX_SURF_FRONT_COLOR(3)
XGL_CTX_SURF_FRONT_COLOR_SELECTOR(3)
XGL_CTX_SURF_FRONT_FILL_STYLE(3)
XGL_CTX_SURF_FRONT_FPAT(3)
XGL_CTX_SURF_FRONT_FPAT_POSITION(3)
XGL_CTX_SURF_INTERIOR_RULE(3)

NOTES

The relationship between supported orders and surface approximation methods for non-rational surfaces is as follows:

Order	Supported Approximation Method
1-10	All methods supported
11-25	XGL_SURF_CONST_PARAM_SUBDIV_BETWEEN_KNOTS only

Non-rational surfaces of order greater than 25 are not supported.

The relationship between supported orders and surface approximation methods for rational surfaces is as follows:

Order	Supported Approximation Method
1-6	All methods supported
7-10	XGL_SURF_CONST_PARAM_SUBDIV_BETWEEN_KNOTS only

Rational surfaces of order greater than 10 are not supported.

The relationship between supported orders and trim curve approximation methods for non-rational trim curves is as follows:

Order	Supported Approximation Method
2-10	All methods supported
11-15	XGL_TRIM_CURVE_CONST_PARAM_SUBDIV_BETWEEN_KNOTS only

The relationship between supported orders and trim curve approximation methods for rational trim curves is as follows:

Order	Supported Approximation Method
2-6	All methods supported
7-15	XGL_TRIM_CURVE_CONST_PARAM_SUBDIV_BETWEEN_KNOTS only

Trim curves of order greater than 15 are not supported.

Rendering of surfaces of high U, V, or trim curve orders (greater than 6) is likely to be very slow.

XGL currently supports the following point types in xgl_nurbs_surface operator:

Xgl_pt_f3d, Xgl_pt_f3h.