NAME

xgl_multiarc – draws a list of arcs

SYNOPSIS

```
#include <xgl/xgl.h>
void  xgl_multiarc (
            Xgl_ctx         ctx,
            Xgl_arc_list    *arc_list );
```

DESCRIPTION

Purpose

This operator draws a list of arcs of a circle according to the graphics state described in the Context *ctx*. Each arc in the list is transformed by the current transformation matrix (see **XGL_CTX_VIEW_TRANS**(3) and **XGL_CTX_MODEL_TRANS**(3)) and drawn in the plane of the view surface in the case of 2D Contexts. When calling xgl_multiarc with a 3D Context, the arcs are rendered in the plane described by the two direction vectors provided with each arc. The result is then projected using the current set of Transforms, so that all of the standard 3D pipeline applies. In the case of 3D arcs, the arcs are approximated as polygons, using the following context attributes:

XGL_CTX_NURBS_CURVE_APPROX(3)
XGL_CTX_NURBS_CURVE_APPROX_VAL(3)
XGL_CTX_MAX_TESSELLATION(3)
XGL_CTX_MIN_TESSELLATION(3)

Parameters

ctx The XGL 2D or 3D Context containing the graphics state used when processing **xgl_multiarc**(3)**.**

arc_list A pointer to the list of arcs to draw. It is a structure defined as:

```
typedef struct {
        Xgl_usgn32          num_arcs;   /* number of arcs */
        Xgl_multiarc_type   type;       /* data type of multiarc */
        Xgl_bbox            *bbox;      /* optional box containing all arcs */
        union {                         /* union of pointers to arcs */
                Xgl_arc_i2d     *i2d;
                Xgl_arc_f2d     *f2d;
                Xgl_arc_d2d     *d2d;
                Xgl_arc_f3d     *f3d;
                Xgl_arc_af3d    *af3d;
                Xgl_arc_d3d     *d3d;
                Xgl_arc_ad3d    *ad3d;
        } arcs;
} Xgl_arc_list;
```

arc_list→num_arcs is the number of arcs in the list.

arc_list→type is the data type of the center point and radius used to define each arc in the list. Xgl_multiarc_type defines the following enumerated values as possible multiarc data types:

XGL_MULTIARC_I2D

XGL_MULTIARC_F2D
XGL_MULTIARC_D2D
XGL_MULTIARC_F3D
XGL_MULTIARC_AF3D.
XGL_MULTIARC_D3D
XGL_MULTIARC_AD3D.

Ultimately, this determines which member of the union *arc_list→arcs* (see below) points to the arc data.

arc_list→bbox points to a structure describing a bounding box for all of the arcs in the list. This may be NULL to indicate that no bounding box information is available.

arc_list→arcs is a union of pointers, one of which points to the array of structures that defines each arc in the list. For example, the *arc_list→arcs* structure for multiarcs of type XGL_MULTIARC_I2D looks as follows:

```
typedef struct {
        Xgl_pt_flag_i2d   center;       /* center point */
        Xgl_usgn32        radius;       /* radius of curvature */
        float             start_angle,  /* start and stop angles */
                          stop_angle;
} Xgl_arc_i2d;
```

The structure for multiarcs of type XGL_MULTIARC_F2D is similar except that the data types of the *center* and *radius* members of the structure become *Xgl_pt_flag_f2d* and *float*, respectively.

The *arc_list→arcs* structure for multiarcs of type XGL_MULTIARC_F3D looks as follows:

```
typedef struct {
        Xgl_pt_flag_f3d   center;           /* center point */
        Xgl_pt_f3d        dir[3];           /* dir. vectors and normal vector */
        Xgl_boolean       dir_normalized;   /* TRUE when dir[] are unit vectors */
        Xgl_boolean       dir_normal;       /* TRUE if dir[2] is a valid vector */
        float             radius;           /* radius of curvature */
        float             start_angle,      /* start and stop angles */
                          stop_angle;
} Xgl_arc_f3d;
```

See **xgl_struct**(3) for structures defining other multiarcs in the list.

The first two elements of the field *dir[3]* contain the two orthogonal vectors in model coordinates (MC) used to define the plane on which the arc is going to be built. The third element is used to specify a normal vector for the arc.

dir_normalized should be set to TRUE when the vectors are normalized (that is, unit vectors). *dir_normal* should be set to TRUE when the third element of *dir* is set to a

valid vector. Otherwise, XGL computes a normal vector to the arc, using the two vectors specified for the arc plane.

When specifying type XGL_MULTIARC_AF3D for multiarcs, XGL generates *annotation arcs*. In other words, the arcs generated by XGL are guaranteed to be in a plane parallel to the projection plane. In this particular case, the center of the arc represents the reference point and is specified in model coordinates. The radius is specified in virtual device coordinates (VDC). There is no need to specify any direction vectors, and the primitive is always front-facing.

Execution

As shown above, each arc is specified by an application-defined center point, radius, and start and stop angle. The *center* and *radius* are specified in the application's coordinate system (except in the case of annotation arcs, as shown before). The *start_angle* is the angle from the positive *x*-axis in the plane of the circle to the start point of the arc. The *stop_angle* is the angle from the positive *x*-axis in the plane of the circle to the end point. The angles are specified in radians. The arc traverses a counterclockwise (CCW) path from start angle to stop angle when the *x*-axis points right and the *y*-axis points up. This path is clockwise (CW) when the *x*-axis points right and the *y*-axis points down. The start and stop angles are defined with these senses, CCW or CW, respectively.

The arcs can be displayed in one of three ''styles,'' depending on the current setting for the attribute **XGL_CTX_ARC_FILL_STYLE**(3)**.** If the arc is closed (that is, if XGL_CTX_ARC_FILL_-STYLE is set to either XGL_ARC_SECTOR or XGL_ARC_CHORD), the following surface-fill attributes control the appearance of the enclosed area of each arc:

XGL_CTX_SURF_FRONT_FILL_STYLE(3)
XGL_CTX_SURF_FRONT_FPAT(3)
XGL_CTX_SURF_FRONT_FPAT_POSITION(3)
XGL_CTX_SURF_FRONT_COLOR(3)

In 3D, the following equivalent attributes for back-facing polygons also apply:

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_COLOR(3)

The lighting and shading attributes are also in effect.

In addition, edges can be drawn around closed arcs. To enable edge drawing, first set the attribute **XGL_CTX_SURF_EDGE_FLAG**(3) to TRUE. Then, edges will be drawn around each arc within the arc list that has set the flag associated with its *center* point. Note that the *center* point of each arc always has a flag associated with it. XGL assumes the application will initialize these flags (to either **0** or **1**) before calling xgl_multiarc. If the flags are not initialized, unpredictable results may occur.

For open arcs (arcs drawn when XGL_CTX_ARC_FILL_STYLE is set to XGL_ARC_OPEN), the line attributes control how the arc is drawn on the Raster. The following attributes are used:

XGL_CTX_LINE_STYLE(3)
XGL_CTX_LINE_PATTERN(3)
XGL_CTX_LINE_WIDTH_SCALE_FACTOR(3)
XGL_CTX_LINE_CAP(3)

XGL_CTX_LINE_JOIN(3)
XGL_CTX_LINE_MITER_LIMIT(3)
XGL_CTX_LINE_COLOR(3)
XGL_CTX_LINE_ALT_COLOR(3)

For a more complete summary of these attributes, see the description for **xgl_multipolyline**(3).

RETURN VALUE

Nothing is returned.

SEE ALSO

xgl_multipolyline(3)
xgl_object_get(3)
xgl_object_set(3)
XGL_CTX_ARC_FILL_STYLE(3)
XGL_CTX_NURBS_CURVE_APPROX(3)
XGL_CTX_NURBS_CURVE_APPROX_VAL(3)
XGL_CTX_MAX_TESSELLATION(3)
XGL_CTX_MIN_TESSELLATION(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)
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)

NOTES

The *center* point of each arc must have the flag associated with it set to either **0** or **1** (to turn the arc's edge either Off or On). If the flag is not set by the application, unpredictable results may occur.

If the difference between the *start_angle* and *stop_angle* of an arc is greater than 2π radians, then an arc is drawn instead of a circle. For example, if *start_angle* is 0.0 radians and *stop_angle* is $(5*\pi/2)$ radians, then an arc is drawn as if the *stop_angle* were $\pi/2$ radians.