NAME

xgl_struct – XGL Data Structures

DESCRIPTION

This section lists the different C structures XGL uses. The structures appear with explanations about the fields used in a given structure.

Arc Structures

In the point given for the center of the arc, the flag field determines whether edges are drawn and also provides storage for an identification number to label individual arcs when picking multiarcs. The start and stop angles are in radians. All 2D angles increase counterclockwise and start (are equal to 0) on the positive *x* axis.

2D Arcs

```
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;

typedef struct {
        Xgl_pt_flag_f2d   center;        /* center point */
        float             radius;        /* radius */
        float             start_angle,   /* start and stop angles */
                          stop_angle;
} Xgl_arc_f2d;

typedef struct {
        Xgl_pt_flag_d2d   center;        /* center point */
        double            radius;        /* radius */
        double            start_angle,   /* start and stop angles */
                          stop_angle;
} Xgl_arc_d2d;
```

3D Arcs

```
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 */
        float             start_angle,       /* start and stop angles */
                          stop_angle;
```

```
} Xgl_arc_f3d;

typedef struct {
        Xgl_pt_flag_f3d   center;           /* center point = ref point in MC */
        float             radius;           /* radius in VDC */
        float             start_angle,      /* start and stop angles */
                          stop_angle;
} Xgl_arc_af3d;

typedef struct {
        Xgl_pt_flag_d3d   center;           /* Center point */
        Xgl_pt_d3d        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 */
        double            radius;           /* Radius */
        double            start_angle,      /* Start and stop angles */
                          stop_angle;
} Xgl_arc_d3d;

typedef struct {
        Xgl_pt_flag_d3d   center;           /* Center point = ref point in MC */
        double            radius;           /* Radius in VDC*/
        double            start_angle,      /* Start and stop angles */
                          stop_angle;
} Xgl_arc_ad3d;                             /* Annotation 3D arc */
```

Arc List Structure

The bounding box applies to all of the arc centers.

```
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;
```

Bounding Box Structures

XGL bounding boxes are used to optimize geometric operations within the XGL pipeline. The application should provide the bounding box information whenever possible. If you don't supply a bbox to function argument, use NULL instead.

```
typedef struct {
        Xgl_bbox_type           bbox_type;/* specifies the type of the bbox */
        union {
                Xgl_bounds_i2d  i2d;   /* type can be integer 2d, */
                Xgl_bounds_f2d  f2d;   /* float 2d, */
                Xgl_bounds_f3d  f3d;   /* or float 3d  */
        } box;
} Xgl_bbox;

typedef struct {
        Xgl_bbox_type           bbox_type;
        union {
                Xgl_bounds_i2d  i2d;
        } box;
} Xgl_bbox_i2d;

typedef struct {
        Xgl_bbox_type           bbox_type;
        union {
                Xgl_bounds_f2d  f2d;
        } box;
} Xgl_bbox_f2d;

typedef struct {
        Xgl_bbox_type           bbox_type;
        union {
                Xgl_bounds_f3d  f3d;
        } box;
} Xgl_bbox_f3d;

typedef struct {
        Xgl_bbox_type           bbox_type;
        union {
                Xgl_geom_status status;
        } box;
} Xgl_bbox_status;

typedef struct {
```

```
        Xgl_bbox_type           bbox_type;
        union {
                Xgl_bounds_d2d  d2d;
        } box;
} Xgl_bbox_d2d;

typedef struct {
        Xgl_bbox_type           bbox_type;
        union {
                Xgl_bounds_d3d  d3d;
        } box;
} Xgl_bbox_d3d;
```

Bounds Structures

Bounds structures are used to define bounding boxes, windows, and viewports or to delimit a range of values between minimal and maximal limits.

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

typedef struct {
        double          bmin;
        double          bmax;
} Xgl_bounds_d1d;

typedef struct {
        float           xmin;
        float           xmax;
        float           ymin;
        float           ymax;
} Xgl_bounds_f2d;

typedef struct {
        double          xmin;
        double          xmax;
        double          ymin;
        double          ymax;
} Xgl_bounds_d2d;

typedef struct {
        float           xmin;
```

```
        float        xmax;
        float        ymin;
        float        ymax;
        float        zmin;
        float        zmax;
} Xgl_bounds_f3d;

typedef struct {
        double       xmin;
        double       xmax;
        double       ymin;
        double       ymax;
        double       zmin;
        double       zmax;
} Xgl_bounds_d3d;

typedef struct {
        Xgl_sgn32    xmin;
        Xgl_sgn32    xmax;
        Xgl_sgn32    ymin;
        Xgl_sgn32    ymax;
} Xgl_bounds_i2d;
```

Circle Structures

In the point given for the center of a circle, the flag field determines whether edges are drawn and also provides storage for an identification number to label individual circles when picking multicircles.

```
typedef struct {
        Xgl_pt_flag_i2d  center;      /* center point */
        Xgl_usgn32       radius;      /* radius */
} Xgl_circle_i2d;

typedef struct {
        Xgl_pt_flag_f2d  center;      /* center point */
        float            radius;      /* radius */
} Xgl_circle_f2d;

typedef struct {
        Xgl_pt_flag_d2d  center;      /* Center point */
        double           radius;      /* Radius */
} Xgl_circle_d2d;
```

```
typedef struct {
        Xgl_pt_flag_f3d  center;          /* center point */
        Xgl_pt_f3d       dir[3];          /* direction and normal vectors */
        Xgl_boolean      dir_normalized;  /* TRUE when dir[] are unit vectors */
        Xgl_boolean      dir_normal;      /* TRUE when dir[3] is a valid vector */
        float            radius;          /* radius */
} Xgl_circle_f3d;

typedef struct {
        Xgl_pt_flag_f3d  center;          /* center point = ref point in MC */
        float            radius;          /* radius in VDC */
} Xgl_circle_af3d;

typedef struct {
        Xgl_pt_flag_d3d  center;          /* Center point */
        Xgl_pt_d3d       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 */
        double           radius;          /* Radius */
} Xgl_circle_d3d;

typedef struct {
        Xgl_pt_flag_d3d  center;          /* Center point = ref point in MC */
        double           radius;          /* Radius in VDC */
} Xgl_circle_ad3d;                        /* Annotation 3D circle */
```

Circle List Structure

The bounding box applies to all of the circle centers.

```
typedef struct {
        Xgl_usgn32             num_circles;   /* Number of circles */
        Xgl_multicircle_type   type;          /* Data type of multicircle */
        Xgl_bbox               *bbox;         /* Optional box containing all
                                                 circles; NULL means no box */
        union {                               /* Union of pointers to circles */
                Xgl_circle_i2d    *i2d;
                Xgl_circle_f2d    *f2d;
                Xgl_circle_d2d    *d2d;
                Xgl_circle_f3d    *f3d;
                Xgl_circle_af3d   *af3d;
                Xgl_circle_d3d    *d3d;
                Xgl_circle_ad3d   *ad3d;
        } circles;
} Xgl_circle_list;
```

Color Structure

The color can be defined as an index, a gray level, an RGB, or a z quadruplet.

```
typedef union {
        Xgl_color_index     index;
        Xgl_color_gray      gray;
        Xgl_color_rgb       rgb;
        Xgl_color_z         z;
} Xgl_color;
```

Color Index Structure

```
typedef   Xgl_usgn32   Xgl_color_index;
```

Color Gray Structure

```
typedef   float        Xgl_color_gray;
```

Color RGB Structure

In RGB, the color is defined by a triplet of floating point values between 0.0 and 1.0.

```
typedef struct {
      float   r, g, b;
} Xgl_color_rgb;
```

Color Z Structure

```
typedef   Xgl_usgn32   Xgl_color_z;
```

Homogeneous Color Structure

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

Homogeneous Color Type

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

Color List Structure

This structure is used to list color entries. It can be used to set an XGL Color Map.

```
typedef struct {
        Xgl_usgn32    start_index;   /* starting index for table */
        Xgl_usgn32    length;        /* number of entries in table */
        Xgl_color     *colors;       /* color table array */
} Xgl_color_list;
```

Color Models Returned by Inquiry

```
typedef struct {
      unsigned      index:1;
      unsigned      rgb:1;
} Xgl_color_type_supported;
```

Color Spline for Color-Mapped NURBS Curves

```
typedef struct {
        Xgl_usgn32                 order;
        Xgl_usgn32                 num_knots;
        float                      *knot_vector;
        Xgl_color_homogeneous      *colors;
} Xgl_curve_color_spline;
```

Elliptical Arc Structures

The elliptical arc structure flag determines whether edges are drawn and also provides storage for an identification number to label individual ellipse when picking multiellipse.

```
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              major_axis,        /* major and minor axis */
                           minor_axis;
        float              rot_angle;         /* ellipse rotation angle */
        float              start_angle,       /* start and stop angles */
                           stop_angle;
} Xgl_ell_f3d;
```

```
typedef struct {
        Xgl_pt_flag_f3d    center;          /* Center point = ref point in MC */
        float              major_axis,      /* Major & Minor axis in VDC*/
                           minor_axis;
        float              rot_angle;       /* Ellipse rotation angle */
        float              start_angle,     /* Start and stop angles */
                           stop_angle;
} Xgl_ell_af3d;                             /* Annotation 3D elliptical arc */

typedef struct {
        Xgl_pt_flag_d3d    center;          /* Center point */
        Xgl_pt_d3d         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 */
        double             major_axis,      /* Major & Minor axis */
                           minor_axis;
        double             rot_angle;       /* Ellipse rotation angle */
        double             start_angle,     /* Start and stop angles */
                           stop_angle;
} Xgl_ell_d3d;

typedef struct {
        Xgl_pt_flag_d3d    center;          /* Center point = ref point in MC */
        double             major_axis,      /* Major & Minor axis in VDC*/
                           minor_axis;
        double             rot_angle;       /* Ellipse rotation angle */
        double             start_angle,     /* Start and stop angles */
                           stop_angle;
} Xgl_ell_ad3d;                             /* Annotation 3D elliptical arc */
```

Elliptical Arc List Structure

```
typedef struct {
        Xgl_usgn32          num_ells;   /* number of elliptical arcs */
        Xgl_multiell_type   type;       /* data type of multiell */
        Xgl_bbox            *bbox;      /* optional box containing all
                                           elliptical arcs, NULL means no box */
        union {                         /* union of pointers to ells */
                Xgl_ell_f3d     *f3d;
                Xgl_ell_af3d    *af3d;
                Xgl_ell_d3d     *d3d;
                Xgl_ell_ad3d    *ad3d;
        } ells;
} Xgl_ell_list;
```

Error Structure Containing Error Attributes

```
typedef struct {
        Xgl_error_type        type;
        Xgl_error_category    category;
        char*                 id;
        char*                 msg;
        char*                 cur_op;
        char*                 cur_obj;
        char*                 operand1;
        char*                 operand2;
} Xgl_error_info;
```

Facet with Color Structure

A facet contains information for a simple polygon, such as a single boundary polygon, or one triangle in a triangle strip, but it can also represent a complex set of boundaries for a planar polygon. The information contained in a facet can be color, normal vector, or both.

```
typedef struct {
    Xgl_color     color;
} Xgl_color_facet;
```

Facet with Normal Vector Structure

For details, see Facet with Color Structure.

```
typedef struct {
    Xgl_pt_f3d     normal;
} Xgl_normal_facet;
```

Facet with Color and Normal Vector Structure

For details, see Facet with Color Structure.

```
typedef struct {
    Xgl_color      color;
    Xgl_pt_f3d     normal;
} Xgl_color_normal_facet;
```

Facet Structure

Used to specify the color or surface normal vector, or both, for one facet.

```
typedef union {
        Xgl_color_facet             color_facet;
        Xgl_normal_facet            normal_facet;
        Xgl_color_normal_facet      color_normal_facet;
} Xgl_facet;
```

List of Polygonal Facets

When providing a list of facet information, all the facets must be of the same type.

```
typedef struct {
        Xgl_facet_type                  facet_type;        /* type of facet */
        Xgl_usgn32                      num_facets;        /* number of facets */
        union {                                            /* pointers to array of facets */
                Xgl_color_facet         *color_facets;
                Xgl_normal_facet        *normal_facets;
                Xgl_color_normal_facet  *color_normal_facets;
        } facets;
} Xgl_facet_list;
```

List of Facet Lists

```
typedef struct {
      Xgl_usgn32     num_facet_lists;    /* number of facet lists */
      Xgl_facet_list *facet_lists;       /* array of facet lists */
} Xgl_facet_list_list;
```

HLHSR Structure

Currently, the only supported HLHSR mode uses a z-buffer. This structure is used to pass initialization information to the z-buffer. Values given for z-value are positive numbers.

```
typedef union {
      struct {
            float   z_value;
      } z_buffer;
} Xgl_hlhsr_data;
```

Inquire Data Structure

```
typedef struct {
          char                      *name;
          Xgl_boolean               dga_flag;
```

```
    Xgl_color_type_supported   color_type;
    Xgl_sgn32                  depth, width, height;
    Xgl_sgn32                  maximum_buffer;
    Xgl_boolean                db_buffer_is_copy;
    Xgl_pt_type_supported      pt_type;
    Xgl_hlhsr_mode             hlhsr_mode;
    Xgl_inq_support_level      picking;
    Xgl_inq_support_level      double_buffer;
    Xgl_inq_support_level      indexed_color;
    Xgl_inq_support_level      true_color;
    Xgl_inq_support_level      depth_cueing;
    Xgl_inq_support_level      lighting;
    Xgl_inq_support_level      shading;
    Xgl_inq_support_level      hlhsr;
    Xgl_inq_support_level      antialiasing;
    Xgl_boolean                stereo;
} Xgl_inquire;
```

Window Rectangles and Window Rectangles List

These structures define the storage type of clip lists used by the window system.

```
typedef struct {
    Xgl_usgn32   xmin;
    Xgl_usgn32   xmax;
    Xgl_usgn32   ymin;
    Xgl_usgn32   ymax;
} Xgl_irect;

typedef struct {
    Xgl_usgn32   num_irects;
    Xgl_irect    *rects;
} Xgl_irect_list;
```

Strings and Text Structures

XGL accepts two different ways of describing text. Text can be a **NULL** -terminated C-style string of characters, or it can be coded as a list of mono-encoded character strings.

```
typedef struct {
    Xgl_object      font_obj;       /* Each mono-encoded string has its */
    char            *text;          /* font and C type string of characters */
} Xgl_mono_text;

typedef struct {
    Xgl_usgn32      mono_num;       /* number of mono-encoded strings */
    Xgl_mono_text   *mono_list;     /* array of mono-encoded strings */
```

```
} Xgl_mono_text_list;
```

NURBS Curve and Surface Structures

These structures are used with the XGL curve and surface operators to draw NURBS curves and surfaces.

```
/* Nonuniform B-spline curve (XGL 1.0 and 2.0 definition of Xgl_nurbs_curve) */
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 */
        Xgl_boolean       trim_curve_vis;   /* If this is a trim curve, is
                                               its outline edge drawn? */
} Xgl_nu_bspline_curve;

/* Nonuniform B-spline curve (definition for XGL 3.0 and subsequent releases) */
typedef struct {
        Xgl_usgn32   order;          /* Order of curve */
        Xgl_usgn32   num_knots;      /* Number of knots */
        float        *knot_vector;   /* Knot vector */
        Xgl_pt_list  ctrl_pts;       /* List of control points */
} Xgl_nurbs_curve;

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;
```

NURBS Surface Geometry Description

```
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;
```

NURBS Surface Simple Geometry Structure

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

Object Descriptor

```
typedef union {
     struct {
          Xgl_window_type    type;    /* window type */
          void               *desc;
     } win_ras;
     char      *sfont_name;
     struct {
          Xgl_object          raster;
     } accum_buf;
     struct {
          char               *name;   /* library name (eg. xglSUNWcgm) */
          void               *desc;   /* opaque handle for stream-specific info. */
     } stream;
} Xgl_obj_desc;
```

Pick Info Structure

When picking graphical entities, the pick IDs are stored into pick info structures. *vertex_flag.*

```
typedef struct {
        Xgl_usgn32    id1;
        Xgl_usgn32    id2;
        Xgl_usgn32    vertex_flag;
} Xgl_pick_info;
```

Plane Structure

```
typedef struct {
        Xgl_pt_d3d    pt;         /* a point on the plane in WC */
        Xgl_pt_d3d    normal;     /* normal vector components */
} Xgl_plane;
```

Plane List Structure

```
typedef struct {
      Xgl_usgn32    num_planes;    /* number of planes */
      Xgl_plane     *planes;       /* pointer to array of planes */
} Xgl_plane_list;
```

Point Structures

XGL defines numerous data types an application can use to specify geometry. Most of these structures define data for a single vertex. A single vertex must at least have the point coordinates in model coordinates. It can also contain color, normal, data, or flag information. The coordinates can be float, double-precision float, or integer (2D). *Color* is defined with an *Xgl_color* structure; the *normal* vector is composed of three floating-point values representing a *normalized vector;* and, in the current implementation, the two lower bits of the *flag* field are used to specify whether an edge should be drawn. The 16 upper bits of the flag field can be used by the application to set an identifier on a per-vertex basis. XGL then returns the identifier vertex information to the application when picking is performed.

A single point structure with only position information.

```
typedef struct {
         Xgl_pt_type        pt_type;     /* type of point */
         Xgl_pt_position    pt;          /* union of pointers to one point */
} Xgl_pt;
```

Structures for Floating Point Coordinates

```
typedef struct {
        float           x;
        float           y;
} Xgl_pt_f2d;

typedef struct {
        float           x;
        float           y;
        Xgl_color       color;
} Xgl_pt_color_f2d;

typedef struct {
        float           x;
        float           y;
        Xgl_usgn32      flag;
} Xgl_pt_flag_f2d;

typedef struct {
        float           x;
        float           y;
        float           w;
} Xgl_pt_f2h;

typedef struct {
        float           x;
        float           y;
        float           z;
} Xgl_pt_f3d;

typedef struct {
        float           x;
        float           y;
        float           z;
        Xgl_color       color;
} Xgl_pt_color_f3d;

typedef struct {
        float           x;
        float           y;
        float           z;
        Xgl_pt_f3d      normal;
```

```
} Xgl_pt_normal_f3d;

typedef struct {
        float           x;
        float           y;
        float           z;
        Xgl_color       color;
        Xgl_pt_f3d      normal;
} Xgl_pt_color_normal_f3d;

typedef struct {
        float           x;
        float           y;
        float           z;
        Xgl_usgn32      flag;
} Xgl_pt_flag_f3d;

typedef struct {
        float           x;
        float           y;
        float           z;
        Xgl_color       color;
        Xgl_usgn32      flag;
} Xgl_pt_color_flag_f3d;

typedef struct {
        float           x;
        float           y;
        float           z;
        Xgl_pt_f3d      normal;
        Xgl_usgn32      flag;
} Xgl_pt_normal_flag_f3d;

typedef struct {
        float           x;
        float           y;
        float           z;
        Xgl_color       color;
        Xgl_pt_f3d      normal;
        Xgl_usgn32      flag;
} Xgl_pt_color_normal_flag_f3d;

typedef struct {
```

```
        float       x;
        float       y;
        float       z;
        float       data[1];
} Xgl_pt_data_f3d;

typedef struct {
        float       x;
        float       y;
        float       z;
        Xgl_color   color;
        float       data[1];
} Xgl_pt_color_data_f3d;

typedef struct {
        float       x;
        float       y;
        float       z;
        Xgl_pt_f3d  normal;
        float       data[1];
} Xgl_pt_normal_data_f3d;

typedef struct {
        float       x;
        float       y;
        float       z;
        Xgl_color   color;
        Xgl_pt_f3d  normal;
        float       data[1];
} Xgl_pt_color_normal_data_f3d;

typedef struct {
        float       x;
        float       y;
        float       z;
        Xgl_usgn32  flag;
        float       data[1];
} Xgl_pt_flag_data_f3d;

typedef struct {
        float       x;
        float       y;
        float       z;
```

```
        Xgl_color     color;
        Xgl_usgn32    flag;
        float         data[1];
} Xgl_pt_color_flag_data_f3d;

typedef struct {
        float         x;
        float         y;
        float         z;
        Xgl_pt_f3d    normal;
        Xgl_usgn32    flag;
        float         data[1];
} Xgl_pt_normal_flag_data_f3d;

typedef struct {
        float         x;
        float         y;
        float         z;
        Xgl_color     color;
        Xgl_pt_f3d    normal;
        Xgl_usgn32    flag;
        float         data[1];
} Xgl_pt_color_normal_flag_data_f3d;

typedef struct {
        float         x;
        float         y;
        float         z;
        float         w;
} Xgl_pt_f3h;
```

Structures for Double-Precision Floating Point Coordinates

```
typedef struct {
        double        x;
        double        y;
} Xgl_pt_d2d;

typedef struct {
        double        x;
        double        y;
        Xgl_color     color;
} Xgl_pt_color_d2d;
```

```
typedef struct {
        double          x;
        double          y;
        Xgl_usgn32      flag;
} Xgl_pt_flag_d2d;

typedef struct {
        double          x;
        double          y;
        double          w;
} Xgl_pt_d2h;

typedef struct {
        double          x;
        double          y;
        double          z;
} Xgl_pt_d3d;

typedef struct {
        double          x;
        double          y;
        double          z;
        Xgl_color       color;
} Xgl_pt_color_d3d;

typedef struct {
        double          x;
        double          y;
        double          z;
        Xgl_pt_f3d      normal;
} Xgl_pt_normal_d3d;

typedef struct {
        double          x;
        double          y;
        double          z;
        Xgl_color       color;
        Xgl_pt_f3d      normal;
} Xgl_pt_color_normal_d3d;

typedef struct {
        double          x;
        double          y;
```

```
        double       z;
        Xgl_usgn32   flag;
} Xgl_pt_flag_d3d;

typedef struct {
        double       x;
        double       y;
        double       z;
        Xgl_color    color;
        Xgl_usgn32   flag;
} Xgl_pt_color_flag_d3d;

typedef struct {
        double       x;
        double       y;
        double       z;
        Xgl_pt_f3d   normal;
        Xgl_usgn32   flag;
} Xgl_pt_normal_flag_d3d;

typedef struct {
        double       x;
        double       y;
        double       z;
        Xgl_color    color;
        Xgl_pt_f3d   normal;
        Xgl_usgn32   flag;
} Xgl_pt_color_normal_flag_d3d;

typedef struct {
        double       x;
        double       y;
        double       z;
        double       w;
} Xgl_pt_d3h;
```

Structures for Integer Point Coordinates

```
typedef struct {
        Xgl_sgn32    x;
        Xgl_sgn32    y;
} Xgl_pt_i2d;
```

```
typedef struct {
        Xgl_sgn32     x;
        Xgl_sgn32     y;
        Xgl_color     color;
} Xgl_pt_color_i2d;

typedef struct {
        Xgl_sgn32     x;
        Xgl_sgn32     y;
        Xgl_usgn32    flag;
} Xgl_pt_flag_i2d;

typedef struct {
        Xgl_sgn32     x;
        Xgl_sgn32     y;
        Xgl_sgn32     w;
} Xgl_pt_i2h;
```

Point List and Point List List Structures

Each list of points describes a single set of connected lines, or a single boundary for polygons. For each list, the application can specify a bounding box in order to optimize geometrical computations. The point type should be identical for all the point lists passed to a single primitive at the same time.

```
typedef struct {
        Xgl_pt_type        pt_type;           /* type of point */
        Xgl_bbox           *bbox;             /* optional box containing
                                                 all points */
        Xgl_usgn32         num_pts;           /* number of points */
        Xgl_usgn32         num_data_values;   /* number of data values at
                                                 each point */
        Xgl_pt_ptr_union   pts;               /* union of pointers to array
                                                 of points */
} Xgl_pt_list;

typedef struct {
        Xgl_bbox      *bbox;
        Xgl_usgn32    num_pt_lists;
        Xgl_pt_list   *pt_lists;
} Xgl_pt_list_list;
```

Union of Pointers to Point Structures

These structures are used as pointers to any XGL point type.

```
typedef union {
        Xgl_pt_i2d                          *i2d;
        Xgl_pt_color_i2d                    *color_i2d;
        Xgl_pt_flag_i2d                     *flag_i2d;
        Xgl_pt_i2h                          *i2h;

        Xgl_pt_f2d                          *f2d;
        Xgl_pt_color_f2d                    *color_f2d;
        Xgl_pt_flag_f2d                     *flag_f2d;
        Xgl_pt_f2h                          *f2h;
        Xgl_pt_f3d                          *f3d;
        Xgl_pt_color_f3d                    *color_f3d;
        Xgl_pt_normal_f3d                   *normal_f3d;
        Xgl_pt_color_normal_f3d             *color_normal_f3d;
        Xgl_pt_flag_f3d                     *flag_f3d;
        Xgl_pt_color_flag_f3d               *color_flag_f3d;
        Xgl_pt_normal_flag_f3d              *normal_flag_f3d;
        Xgl_pt_color_normal_flag_f3d        *color_normal_flag_f3d;
        Xgl_pt_f3h                          *f3h;
        Xgl_pt_data_f3d                     *data_f3d;
        Xgl_pt_color_data_f3d               *color_data_f3d;
        Xgl_pt_normal_data_f3d              *normal_data_f3d;
        Xgl_pt_color_normal_data_f3d        *color_normal_data_f3d;
        Xgl_pt_flag_data_f3d                *flag_data_f3d;
        Xgl_pt_color_flag_data_f3d          *color_flag_data_f3d;
        Xgl_pt_normal_flag_data_f3d         *normal_flag_data_f3d;
        Xgl_pt_color_normal_flag_data_f3d   *color_normal_flag_data_f3d;

        Xgl_pt_d2d                          *d2d;
        Xgl_pt_color_d2d                    *color_d2d;
        Xgl_pt_flag_d2d                     *flag_d2d;
        Xgl_pt_d2h                          *d2h;
        Xgl_pt_d3d                          *d3d;
        Xgl_pt_color_d3d                    *color_d3d;
        Xgl_pt_normal_d3d                   *normal_d3d;
        Xgl_pt_color_normal_d3d             *color_normal_d3d;
        Xgl_pt_flag_d3d                     *flag_d3d;
        Xgl_pt_color_flag_d3d               *color_flag_d3d;
        Xgl_pt_normal_flag_d3d              *normal_flag_d3d;
        Xgl_pt_color_normal_flag_d3d        *color_normal_flag_d3d;
        Xgl_pt_d3h                          *d3h;
        Xgl_pt_data_d3d                     *data_d3d;
} Xgl_pt_ptr_union;
```

Union of Pointers to Points with Only Position Information

```
typedef union {
        Xgl_pt_i2d    *i2d;
        Xgl_pt_f2d    *f2d;
        Xgl_pt_d2d    *d2d;
        Xgl_pt_f3d    *f3d;
        Xgl_pt_d3d    *d3d;
} Xgl_pt_position;
```

Point Types Returned by Inquiry

```
typedef struct {
        unsigned    pt_dim_2d:1;
        unsigned    pt_dim_3d:1;
        unsigned    pt_type_int:1;
        unsigned    pt_type_float:1;
        unsigned    pt_type_double:1;
} Xgl_pt_type_supported;
```

Rectangle Structures

In the point given for the first point of the rectangle, the flag field determines whether edges are drawn and also provides storage for an identification number to label individual rectangles when picking multirectangles.

```
typedef struct {
        Xgl_pt_flag_i2d    corner_min;
        Xgl_pt_i2d         corner_max;
} Xgl_rect_i2d;

typedef struct {
        Xgl_pt_flag_f2d    corner_min;
        Xgl_pt_f2d         corner_max;
} Xgl_rect_f2d;

typedef struct {
        Xgl_pt_flag_d2d    corner_min;
        Xgl_pt_d2d         corner_max;
} Xgl_rect_d2d;

typedef struct {
        Xgl_pt_flag_f2d    corner_min;    /* 1st corner in the plane */
        Xgl_pt_f2d         corner_max;    /* 2nd corner in the plane */
        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 */
} Xgl_rect_f3d;

typedef struct {
        Xgl_pt_flag_f3d    corner_min;       /* 1st corner = ref point in MC */
        Xgl_pt_f2d         corner_max;       /* deltas x&y in VDC, rel. to 1st point */
} Xgl_rect_af3d;

typedef struct {
        Xgl_pt_flag_d3d    corner_min;       /* reference corner of the rectangle */
        Xgl_pt_d2d         corner_max;       /* width & height of the rectangle, in MC */
        Xgl_pt_d3d         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 */
} Xgl_rect_d3d;

typedef struct {
        Xgl_pt_flag_d3d    corner_min;       /* 1st corner = ref point in MC */
        Xgl_pt_d2d         corner_max;       /* deltas x&y in VDC, rel. to 1st point */
} Xgl_rect_ad3d;                             /* Annotation 3D rectangle */
```

Rectangle List Structure

The bounding box applies to all the rectangles.

```
typedef struct {
        Xgl_usgn32           num_rects;   /* number of rectangles */
        Xgl_multirect_type   rect_type;   /* data type of multirectangle */
        Xgl_bbox             *bbox;       /* optional box contain. all rectangles */
        union {                           /* union of pointers to rectangles */
                Xgl_rect_i2d    *i2d;
                Xgl_rect_f2d    *f2d;
                Xgl_rect_d2d    *d2d;
                Xgl_rect_f3d    *f3d;
                Xgl_rect_af3d   *af3d;
                Xgl_rect_d3d    *d3d;
                Xgl_rect_ad3d   *ad3d;
        } rects;
} Xgl_rect_list;
```

Color Map Ramp Structure

An XGL color ramp is defined by an offset, where the ramp starts in the color_map, and by the length of the ramp.

```
typedef struct {
        Xgl_usgn32    offset;    /* starting index of ramp in color table */
        Xgl_usgn32    length;    /* length of ramp */
} Xgl_segment;
```

Color Spline for Color-Mapped NURBS Surfaces

```
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;
```

Texture Operation Structures and Unions

```
typedef struct {
        Xgl_color_rgb                 c1;
        Xgl_color_rgb                 c2;
} Xgl_texture_blend_rgb;

typedef union {
        Xgl_texture_blend_rgb         rgb;
} Xgl_texture_blend_op;

typedef struct {
        Xgl_color_rgb                 c;
} Xgl_texture_decal_rgb;

typedef union {
        Xgl_texture_decal_rgb         rgb;
} Xgl_texture_decal_op;

typedef struct {
        Xgl_color_rgb                 c;
} Xgl_texture_blend_intrinsic_rgb;
```

```
typedef union {
        Xgl_texture_blend_intrinsic_rgb     rgb;
} Xgl_texture_blend_intrinsic_op;

typedef struct {
        Xgl_render_component                comp;
        Xgl_texture_op                      texture_op;
        union Xgl_op_info {
                Xgl_texture_blend_op        blend;
                Xgl_texture_decal_op        decal;
                Xgl_texture_blend_intrinsic_op  blend_int;
        } op;
} Xgl_render_component_desc;
```

Description for Texture Map Types

```
typedef struct {
        Xgl_texture_type            texture_type
        Xgl_texture_color_comp_info comp_info;
        Xgl_texture_interp_info     interp_info;  /* filter to use */
        union {
                Xgl_texture_mipmap_desc   mipmap;
        } info;
} Xgl_texture_desc;

typedef union {
        Xgl_texture_color_comp_info         color_info;
} Xgl_texture_comp_info;

typedef struct {
        Xgl_texture_type            texture_type;
        Xgl_texture_comp_info       comp_info;
        union {
                Xgl_texture_general_mipmap_desc   mipmap;
        } texture_info;
} Xgl_texture_general_desc;
```

Texture Map Interpolation Information

```
typedef struct {
        Xgl_texture_interp_method    filter1;
        Xgl_texture_interp_method    filter2;
} Xgl_texture_interp_info;
```

```
typedef struct {
        Xgl_usgn32                     num_render_comp_desc;
        Xgl_render_component_desc      render_component_desc[2];
        Xgl_usgn32                     num_channels[2];
        Xgl_usgn32                     channel_number[2];
} Xgl_texture_color_comp_info;
```

Texture MipMap Description

```
typedef struct {
        Xgl_texture              texture_map;
        float                    max_u_freq;
        float                    max_v_freq;
        Xgl_texture_boundary     u_boundary;
        Xgl_texture_boundary     v_boundary;
        Xgl_usgn8                boundary_values[4];
        float                    depth_interp_factor;
} Xgl_texture_mipmap_desc;
```

```
typedef struct {
        Xgl_texture               texture_map;
        float                     max_u_freq;
        float                     max_v_freq;
        Xgl_texture_boundary      u_boundary;
        Xgl_texture_boundary      v_boundary;
        Xgl_usgn8                 boundary_values[4];
        float                     depth_interp_factor;
        Xgl_texture_interp_info   interp_info;
        Xgl_matrix_f2d            orientation matrix;
} Xgl_texture_general_mipmap_desc;
```

Threshold Structure

Individual values can be given for primitives rendered with vectors (polylines, curves, stroke text), circles, and polygons.

```
typedef struct {
      Xgl_usgn32     vectors;
      Xgl_usgn32     circles;
      Xgl_usgn32     polygons;
} Xgl_threshold;
```

NURBS Trim Curve and Trim Loop List Structures

The trim curve and loop structures define the trimmed portion of the displayed NURBS surface.

```
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;

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

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

X11 Window Descriptor Structure

X11 data structure for create raster call with XGL_WIN_X. Please refer to example programs on how to fill the fields of this structure.

```
typedef struct {
        void            *X_display;     /* pointer to X display */
        int             X_screen;       /* X screen number */
        Xgl_usgn32      X_window;       /* X window or drawable from server */
} Xgl_X_window;
```

SEE ALSO

xgl_enum_types(3)
xgl_macro_values(3)
xgl_pt_list(3)