NAME

xgl_transform_point – transforms a single point in place

SYNOPSIS

```
#include <xgl/xgl.h>

void  xgl_transform_point (
            Xgl_trans    trans,
            Xgl_pt       *point );
```

DESCRIPTION

Purpose

This operator transforms a single point. The operator treats the point as a row vector and multiplies it by a Transform's matrix. The result is stored back in the memory allocated for the point, overwriting the untransformed point. If the transformation results in a point with a homogeneous *w* coordinate not equal to 1, the *x*, *y*, and *z* (for 3D) coordinates are normalized by *w* to produce an ordinary point with *w* = 1. The dimension of the Transform as given by **XGL_TRANS_DIMENSION**(3) must match the dimension of the point. A 2D point must be given with a 2D Transform. Similarly, a 3D point must be given with a 3D Transform. The data type of the Transform as given by **XGL_TRANS_DATA_TYPE**(3), however, can be different from that of the point.

Parameters

trans The Transform whose matrix is applied in transforming the point. It must have been created previously with **xgl_object_create**(3).

point A pointer to a structure of type *Xgl_pt* defined as follows:

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

point→pt_type is the type of the point in *point→pt*. The table below gives the valid point types.

point→pt is a union of pointers. The appropriate pointer, as determined by *point→pt_type*, gives the address of a point structure containing coordinates. The valid members of this union are as follows:

Field Name	Type	Point Type
pt.i2d	Xgl_pt_i2d *	XGL_PT_I2D
pt.f2d	Xgl_pt_f2d *	XGL_PT_F2D
pt.d2d	Xgl_pt_d2d *	XGL_PT_D2D
pt.f3d	Xgl_pt_f3d *	XGL_PT_F3D
pt.d3d	Xgl_pt_d3d *	XGL_PT_D3D

RETURN VALUE

Nothing is returned.

SEE ALSO

xgl_object_create(3)
XGL_TRANS_DATA_TYPE(3)
XGL_TRANS_DIMENSION(3)

NOTES

The dimensions of the point and the Transform must be the same, either 2D or 3D.