NAME

XGL_TMAP_DESCRIPTOR – specifies the properties of the texture map image

SYNOPSIS

```
#include <xgl/xgl.h>
Xgl_tmap        tmap;
xgl_object_set(tmap, XGL_TMAP_DESCRIPTOR, Xgl_texture_general_desc, 0);
xgl_object_get(tmap, XGL_TMAP_DESCRIPTOR, Xgl_texture_general_desc *);
```

DESCRIPTION

This attribute specifies the list of *textures* associated with this Texture Map object. The *textures* are used when rendering XGL 3D surface primitives.

XGL_TMAP_DESCRIPTOR is a pointer to a C structure of type *Xgl_texture_general_desc*, which is defined as follows:

```
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;
```

Parameters

texture_type

Each texture descriptor has a *texture_type*. Currently, Xgl_texture_type defines XGL_TEXTURE_TYPE_MIPMAP as the only possible value.

comp_info

This parameter explains how different channels in the texture map are used to affect the rendering. The *Xgl_texture_comp_info* structure is defined as:

```
typedef union {
        Xgl_texture_color_comp_info             color_info;
} Xgl_texture_comp_info;
```

The *Xgl_texture_color_comp_info* structure is defined as:

```
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;
```

This structure can have a maximum of 2 render components corresponding to color and alpha. The *num_render_comp_desc* parameter gives the number of render components used by this particular Texture Map object. The first element of the array for *render_component_desc, num_channels* and *channel_number* correspond to values for color component, and the second element of the three arrays correspond to the alpha component. The value of the num_channels[0] element can either be 1 or 3. If the value of the num_channels[0] element is 1,

then the corresponding value in channel_number[0] determines the channel number in the texture image used in composing color; since alpha can use only 1 channel, num_channel[1] is always 1 and channel_number[1] determines the channel number in the texture image used in composing alpha. For example, if num_channels[0] is 1 and if channel_number[0] is 0, then it implies that channel number 0 is of the texture image used in composing the color.

The *Xgl_render_component_desc* structure is defined as:

```
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;
```

The *comp* parameter determines which lighting component in the rendering pipeline (diffuse, reflected, and final-after depth cueing) is affected by this particular texture description. The *comp* field is only valid while composing color since alpha values are unaffected by lighting or depth cueing. Therefore, this field is ignored for *render_component_desc[1].* The *texture_op* parameter specifies how the current color and alpha values are combined with the value obtained from the texture map. The table below lists the valid combinations:

Texture Operations	num_channel 1	num_channel 3
Replace	$\alpha_f = \alpha_t$	$C_f = C_t$
Modulate	$C_f = S_t C_i$ $\alpha_f = \alpha_t * \alpha_i$	$C_f = C_t C_i$
Decal	$\alpha_f = \alpha_i$	$C_f = (1-\alpha_{t'})*$ $C_{k1} + \alpha_{t'} * C_{t'}$
Blend	$C_f = C_{k1} * (1-S_t) + C_{k2} * S_t$ $\alpha_f = \alpha_t * \alpha_i$	$C_f = C_{k1} * (1-C_t) + C_{k2} * C_t$
Decal Intrinsic	$\alpha_f = \alpha_i$	$C_f = C_i * (1-\alpha_{t'}) +$ $\alpha_{t'} * C_{t'}$
Blend Intrinsic	$C_f = C_i * (1-S_t) +$ $C_{k1} * S_t$ $\alpha_f = \alpha_t * \alpha_i$	$C_f = C_i * (1-C_t) +$ $C_{k1} * C_t$

Where: C_{k1} = Constant color 1
C_{k2} = Constant color 2
C_i = Incoming color
C_f = Final color
S_t = A single channel texture image as specified by channel_number[0]

C_t = Three channels of a texture image
α_t = A single channel from texture image as specified by channel_number[1]
α_f = Final alpha
α_i = Incoming alpha
$C_{t'}$ = First 3 channels of a 4 channel texture image
$\alpha_{t'}$ = Fourth channel of a 4 channel texture image

When *texture_op* is "decal" and if 4 channels are present in the texture image, the decal operation is performed. If only 3 channels are present in the texture image, then a replace is performed.

Note: Blank fields indicate undefined values. The values from the texture image are in the range [01]. Currently in XGL, a 3 channel texture image is specified as a 32-bit value. The upper byte is ignored for 3 channel texture operations.

Texture operations Blend, Decal, and Blend_intrinsic use values defined in the *Xgl_op_info* union. The blending operation gets the constant colors C_{k1} and C_{k2} from the structure *Xgl_texture_blend_op*. The *decal* and *blend_int* operations get the constant color C_{k1} from the structures *Xgl_texture_decal_op* and *Xgl_texture_-blend_intrinsic_op*, respectively.

mipmap This parameter defines the mipmap specific fields. The *Xgl_texture_general_-mipmap_desc* structure is defined as:

```
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;
```

The *texture_map* parameter gives a pointer to the texture map object. A texture map object is created using **xgl_object_create**(3) with XGL_MIPMAP_TEXTURE as the object type.

Note: At this release, the fields *max_u_freq* and *max_v_freq* are not implemented.

The parameters *u_boundary* and *v_boundary* specify the boundary condition to apply when texture bounds are exceeded. The available boundary options are specified by the *Xgl_texture_boundary* structure. The possible values are:

XGL_TEXTURE_BOUNDARY_CLAMP
Finish texturing using a constant value.

XGL_TEXTURE_BOUNDARY_WRAP
Repeat the texture.

XGL_TEXTURE_BOUNDARY_TRANSPARENT
: Finish rendering as though texturing were not being performed.

XGL_TEXTURE_BOUNDARY_MIRROR
: Reverse texel sampling creating a *mirrored alternating* pattern.

XGL_TEXTURE_BOUNDARY_CLAMP_BOUNDARY
: Use a color specified by the closest boundary texel.

Note that if *u* and *v* boundary conditions are not identical, and if both values of *u* and *v* exceed the texture boundary in both directions, the boundary methods will be used in the following order:

(1) XGL_TEXTURE_BOUNDARY_TRANSPARENT

(2) XGL_TEXTURE_BOUNDARY_CLAMP

(3) XGL_TEXTURE_BOUNDARY_WRAP

(4) XGL_TEXTURE_BOUNDARY_MIRROR

(5) XGL_TEXTURE_BOUNDARY_CLAMP_BOUNDARY

The *boundary_values* parameter specifies the value to clamp to when either *u* or *v* value exceeds bounds and when the boundary condition is XGL_TEXTURE_-BOUNDARY_CLAMP. The *depth_interp_factor* parameter is a floating point number that allows the application to adjust the sample level in either direction to obtain a more or less aliased textured surface. A value of 1.0 moves the depth away from the mipmap base creating a blurry event. A value of -1.0 moves the depth towards the mipmap pyramid base for a more detailed picture.

The *interp_info* parameter gives information on the nature of sampling desired by the application. The *Xgl_texture_interp_info* structure is defined as:

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

During the texturing process, the data from the texture mipmap can be sampled or interpolated in a number of ways in order to come up with a value to apply to the pixel. The available methods of sampling and interpolating the mipmap are specified by the *Xgl_texture_interp_method* structure. The possible values are:

Note: The word *texel* used below refers to one element stored in a texture map.

XGL_TEXTURE_INTERP_POINT
: Sample one texel point from the base (top or first) mipmap level at the closest u and v to the pixel.

XGL_TEXTURE_INTERP_BILINEAR
: Sample the four texels closest to the center of the pixel from the base mipmap level and average them.

XGL_TEXTURE_INTERP_MIPMAP_POINT
: Sample one texel point from the nearest mipmap level at the closest u

and v to the pixel.

XGL_TEXTURE_INTERP_MIPMAP_BILINEAR
: Sample the four texels closest to the center of the pixel from the nearest mipmap level and average them.

XGL_TEXTURE_INTERP_MIPMAP_TRILINEAR
: Sample the four texels closest to the center of the pixel from the nearest mipmap level above the depth *d* and the four texels closest from the mipmap level below the depth *d* and average all 8 of them.

XGL_TEXTURE_INTERP_MIPMAP_PT_LINEAR
: Chose the two mipmaps that most closely match the size fo the pixel being textured and find the nearest to the center of the pixel to produce a texture value from each mipmap. The final texture value is an average of those two values.

filter1 is used if the textured pixel maps to an area greater than one texel. *filter2* is used if the textured pixel maps to an area less than or equal to one texel. We strongly recommend that *filter2* only use XGL_TEXTURE_INTERP_POINT or XGL_TEXTURE_INTERP_BILINEAR.

The *orientation_matrix* parameter defines a matrix by which *textures* can be translated and/or rotated. This parameter specifies a 3×2 matrix by which textures can be rotated or translated across the face of the polygon by transforming the *u* and *v* values through the matrix. The *u* and *v* values are specified as data values in the point list of a surface primitive (for example, **xgl_multi_simple_polygon**(3), **xgl_triangle_strip**(3), etc.) if **XGL_TMAP_COORD_SOURCE**(3) is set to XGL_TEXTURE_COORD_DATA. The *u*, *v* values defined in the point list are subject to the **XGL_TMAP_T0_INDEX**(3) and **XGL_TMAP_T1_INDEX**(3) attributes before the orientation matrix is applied.

The default value is NULL.

SEE ALSO

xgl_object_set(3)
xgl_object_get(3)
xgl_object_create(3)
xgl_multi_simple_polygon(3)
xgl_polygon(3)
xgl_quadrilateral_mesh(3)
xgl_triangle_list(3)
xgl_triangle_strip(3)
XGL_TMAP_COORD_SOURCE(3)
XGL_TMAP_T0_INDEX(3)
XGL_TMAP_T1_INDEX(3)

NOTES

Texturing is only supported when the color type of the raster attached to a 3D Context is XGL_COLOR_RGB.