NAME

XGL_DMAP_TEXTURE_DESCRIPTORS – specifies list of *textures* in a Data Map Texture object

SYNOPSIS

#include <xgl/xgl.h>

Xgl_dmap_texture dmap_texture;

xgl_object_set(dmap_texture, XGL_DMAP_TEXTURE_DESCRIPTORS, Xgl_texture_desc, 0);
xgl_object_get(dmap_texture, XGL_DMAP_TEXTURE_DESCRIPTORS, Xgl_texture_desc ∗);

DESCRIPTION

This attribute specifies the list of *textures* associated with this Data Map Texture object. The *textures* are used when rendering XGL 3D surface primitives. When a data map texture has multiple texture descriptions, they are applied sequentially and are cumulative.

The application uses this attribute in conjunction with the attribute **XGL_DMAP_TEXTURE_-NUM_DESCRIPTORS**(3) to specify how many *textures* should be considered when rendering XGL primitives.

Setting a new list of *textures* replaces the previous (if any) list of *textures*. The application is also responsible for making sure that enough memory is available when getting the list of *textures* from XGL.

XGL_DMAP_TEXTURE_DESCRIPTORS is a pointer to a C structure of type *Xgl_texture_desc*, which is defined as follows:

```
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;
```

Parameters

texture_type

Each texture descriptor has a *texture_type*. Currently, XGL_TEXTURE_TYPE_MIPMAP is the only possible value for this attribute.

comp_info

This parameter explains how different channels in the texture map are used to affect the rendering. 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. The *num_render_-comp_desc* parameter gives the number of render components used by this particular Texture Map object. Note that in the current release of XGL, only *num_render_comp_desc* of 1 is supported. The *num_channels[2]* parameter gives the number of channel used to do color composition. The *channel_number[2]* indicates the channel number used when *num_channel* is 1. 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 *texture_op* parameter specifies how the current color value is combined with the value obtained from the texture map. The table below lists the valid combinations:

Texture Operations

# of Channel	Replace	Modulate	Decal	Blend	Decal Intrinsic	Blend Intrinsic
1		$C_f = S_t C_i$		$C_f = C_{k1} * (1 - S_t) + C_{k2} * S_t$		$C_f = C_i * (1 - S_t) + C_{k1} * S_t$
2		$C_f = S_t C_i$		$C_f = C_{k1} * (1 - S_t) + C_{k2} * S_t$		$C_f = C_i * (1 - S_t) + C_{k1} * S_t$
3	$C_f = C_t$	$C_f = C_t C_i$		$C_f = C_{k1} * (1 - C_t) + C_{k2} * C_t$		$C_f = C_i * (1 - C_t) + C_{k1} * C_t$
4	$C_f = C_t$	$C_f = C_t C_i$	$C_f = (1 - \alpha_t) * C_{k1} + \alpha_t * C_t$	$C_f = C_{k1} * (1 - C_t) + C_{k2} * C_t$	$C_f = C_i * (1 - C_t) + \alpha_t * C_t$	$C_f = C_i * (1 - C_t) + C_{k1} * C_t$

Where: C_{k1} = Constant color 1
C_{k2} = Constant color 2
C_i = Incoming color
C_f = Final color
S_t = Single channel texture image
C_t = First three channels of a 4 channel texture image
α_t = Second channel of a 2 channel texture image or
Fourth channel of a 4 channel texture image

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.

In the *Xgl_op_info* union, the texture operations *blend*, *decal*, and *blend_int* use values defined in the structures *Xgl_texture_blend_op*, *Xgl_texture_decal_op*, and *Xgl_texture_blend_intrinsic_op*, respectively. 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.

interp_info

This 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.

mipmap This parameter defines the mipmap specific fields. The *Xgl_texture_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_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. 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 default value is NULL.

SEE ALSO

xgl_object_set(3)
xgl_object_get(3)
xgl_object_create(3)
XGL_DMAP_TEXTURE_NUM_DESCRIPTORS(3)

NOTES

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

At this release, a new object, Texture Map, has been provided for texture mapping. The Data Map Texture object is retained for backward compatibility, but will be removed from the XGL library in a future release. Because of this, we recommend that application programs use the Texture Map object for texture mapping.