Graficación por Computadora

OpenGL

Mapeo de texturas

Las texturas son una parte importante que nos ayuda a dar realismo a una escena, a un costo aceptable.

Texturizar una superficie consiste esencialmente en pintar una imagen en ella.

Texturas

OpenGL

Mapeo de texturas

Este proceso se puede usar para obtener las siguientes ventajas:

-Autenticidad.

-Ilusión de detalles geométricos

Texturas

OpenGL

Mapeo de texturas

Autenticidad

Representación realista de objetos, ejemplo una caja de cereal, que requiere ser pintada en la realidad, requiere que se pinte la superficie que modela el objeto.

Texturas

OpenGL

Mapeo de texturas

Ilusión de detalles geométricos

En lugar de reproducir la geometría de un objeto, podemos pintar una imagen de la geometría en la escena, esto nos da un resultado realista en una fracción del costo en numero de polígonos

Texturas

OpenGL

Mapeo de texturas

Hay varias maneras en las que se aplica una textura en OpenGL.

El mas común es donde la textura es una arreglo 2D de píxeles

Mapeo de texturas

Las texturas en si pueden ser una imagen externa la cual se importa en el programa o una creada por el programa en si.

La primera se le llama una textura externa mientras que a la segunda se la llama textura procedural.

Texturas

OpenGL

Mapeo de texturas

Los pasos necesarios para realizar el mapeo de texturas son:

1. Crear un objeto textura y especificar una textura para ese objeto.

2. Indicar como se aplica la textura a cada píxel

3. Habilitar el mapeo de texturas.

4. Dibujar la escena.

Mapeo de texturas

Nota: Los píxeles de una textura se les llama texeles.

Texturas

OpenGL

Mapeo de texturas

Crear un objeto textura y especificar una textura para ese objeto.

Una textura es usualmente dos dimensional, pero puede haber también uno y tres dimensional.

Los datos que describen una textura pueden consistir de 1 a 4 elementos por texel y pueden representar una quadrupla (R,G,B,A).

Texturas

OpenGL

Mapeo de texturas

Indicar como se aplica la textura a cada píxel

Se pueden escoger funciones para calcular el valor final RGBA del color y los datos de la textura.

Ejemplo:

-Replace.

-Modulate.

Mapeo de texturas

Habilitar el mapeo de texturas.

Necesitamos habilitar el texturizado antes del dibujado.

Para esto usamos glEnable() con el parámetro correspondiente.La deshabilitamos con glDisable() .

Mapeo de texturas

Habilitar el mapeo de texturas.

Parámetros disponibles:

GL_TEXTURE_1D
GL_TEXTURE_2D
GL_TEXTURE_3D
GL_TEXTURE_CUBE_MAP

Mapeo de texturas

Dibujar la escena.

Se requiere especificar las coordenadas de la textura y las coordenadas geometricas.

Mapeo de texturas

Antes de crear las texturas necesitamos especificarlas para eso usamos glGenTextures() y glBindTextures().

Mapeo de texturas

- void glGenTextures(GLsizei n, Glunit * textures)

n representa los nombres no usados actualmente para objetos texturas en el arreglo textures.

Cero es reservado y nunca se regresa como un nombre de textura.

Mapeo de texturas

- void glBindTexture(GLenum target, Glunit texture)

target es tipo de textura que son

GL_TEXTURE_1D,
GL_TEXTURE_2D,
GL_TEXTURE_3D o
GL_TEXTURE_CUBE_MAP

Texture especifica el nombre de la textura.

Mapeo de texturas

Tambien tenemos el metodo:

- GLBoolean glIsTexture(GLuni texture)

texture el nombre de la textura a verificar.

Mapeo de texturas

Como vamos ligando y desligando objetos textura, sus datos permanecen en los recursos de textura.

Si estos recursos son limitados, el borrado de texturas es necesario.

Mapeo de texturas

Para llevar a cabo el borrado usamos:

-void glDeleteTexture(GLsizei n, const Glunit * texture)

n el numero de objetos textura nombrados por el arreglo.

texture el arreglo con los nombres.

Mapeo de texturas

El comando glTexture* define una textura n-dimensional con $1 \leq n \leq 3$

Para 2D tenemos:

```
void glTexture2D(GLenum target, GLint level,
                 GLint internalFormat, GLsizei width,
                 GLsizei height, GLint border,
                 GLenum format, GLenum type,
                 const GLvoid * texels)
```

Mapeo de texturas

En el código podemos las llamadas a glTexParamter*(), estas especifican como la textura se envuelve y como se filtran los colores sino hay conciencia entre los texeles en la textura y los píxeles en la pantalla

Mapeo de texturas

-void glTextParameter{fi}(GLenum target,
GLenum pname, GLType param)

-void glTextParameter{fi}v(GLenum target,
GLenum pname, GLType *params)

-void glTextParameter{liv luiv}(GLenum target,
GLenum pname, GLType *params)

Texturas

OpenGL

Mapeo de texturas

-target especifica el tipo de textura

-pname especifica uno de los parámetros de la textura

-param o params especifica los valores del parámetro anterior

Mapeo de texturas

En el metodo setup se habilita la textura con glEnable(GL_TEXTURE_2D).

Y en la rutina de dibujado encontramos el comando glTextCoord2f() que nos permite asignar las coordenadas de la textura.

Mapeo de texturas

-void glTextCoord{1234}{sifd}(GLtype s, ...)

-void glTextCoord{1234}{sifd}v(const GLtype * v)

Especificamos las coordenadas de la textura.

Mapeo de texturas

También en el método setup tenemos la rutina glTexEnv*() que asigna el modo de dibujado a replace, de tal manera que los polígonos texturizados usen los colores de la textura.

Mapeo de texturas

- void glTexEnv{fi}v(GLenum target,
  GLenum pname,
  GLint *params)

-target especifica un ambiente de textura.

-pname especifica el parámetro de ambiente.

-params especifica los valores de los parámetros

- Si target es GL_TEXTURE_FILTER_CONTROL entonces pname debe ser GL_LOD_BIAS.

- Si target es GL_TEXTURE_ENV entonces pname puede debe ser: GL_TEXTURE_ENV_MODE,GL_TEXTURE_ENV_COLOR, GL_COMBINE_RGB, GL_COMBINE_ALPHA, GL_RGB_SCALE, GL_ALPHA_SCALE, GL_SRC0_RGB, GL_SRC1_RGB, GL_SRC2_RGB, GL_SRC0_ALPHA, GL_SRC1_ALPHA, or GL_SRC2_ALPHA.

Mapeo de texturas

- Si pname es GL_TEXTURE_ENV_MODE entonces params puede debe ser:
- GL_ADD, GL_MODULATE, GL_DECAL, GL_BLEND, GL_REPLACE, o GL_COMBINE.

Mapeo de texturas

Usando el codigo loadTextures:

Cambiar las coordenadas donde se indica

- (0.0,0.5) (0.5,0.0) (0.5,0.5) (0.0,0.5) textura

- (0.0,0.0) (1.0,0.0) (1.0,1.0) textura
  (-10.0,-10.0,0.0) (10.0,-10.0,0.0) (10.0,10.0,0.0) geometría

Texturas

OpenGL

Mapeo de texturas

Cambiar las coordenadas donde se indica

- (0.0,0.0) (1.0,0.0) (1.0,1.0) textura
  (-10.0,-10.0,0.0) (10.0,-10.0,0.0) (0.0,10.0,0.0) geometría

- (0.0,0.0) (1.0,0.0) (0.5,1.0) textura
  (-10.0,-10.0,0.0) (10.0,-10.0,0.0) (0.0,10.0,0.0) geometría

Texturas

OpenGL

Mapeo de texturas

Usando el codigo loadTextures:

Cambiar la textura lauch.bmp por cray2.bmp.Observar que sucede.

Mapeo de texturas

Cambiar las dimensiones del polígono para evitar la distorsión

(-20.0,-10.0,0.0) (20.0,-10.0,0.0) (20.0,10.0,0.0)
(-20.0,10.0,0.0)

Mapeo de texturas

Cambiar las coordenadas donde se indica

- (0.0,0.0) (1.0,0.0) (1.0,1.0) (0.0,1.0) textura
  (-10.0,-10.0,0.0) (10.0,-10.0,0.0) (20.0,0.0,0.0)
  (-10.0,10.0,0.0) geometría

- (0.0,0.0) (1.0,0.0) (1.0,0.5) (0.5,1.0) (0.0,1.0) textura
  (-10.0,-10.0,0.0) (10.0,-10.0,0.0) (20.0,0.0,0.0)
  (0.0,10.0,0.0) (-10.0,0.0,0.0) geometría

Texturas

OpenGL

Mapeo de texturas

Podemos asignar coordenadas de textura fuera del rango [0,1] para realizar una restricción o repetición en el mapeo de texturas.

Para la restricción cualquier valor mayor a 1 se asigna como 1 y cualquier valor menor que 0 se asigna como 0.

Mapeo de texturas

Cambiar las coordenadas donde se indica

- (-1.0,0.0) (2.0,0.0) (2.0,2.0) (-1.0,2.0) textura

Texturas

OpenGL

Mapeo de texturas

Para poder realizar la repetición de una textura debemos usar glTextParameter con el parámetro GL_TEXTURE_WRAP_* y los valores del parámetro debe ser GL_REPEAT.

Mapeo de texturas

Entonces tenemos:

```
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_S,
                 GL_REPEAT);

glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_T,
                 GL_REPEAT);
```

Mapeo de texturas

También tenemos la opción GL_MIRRORED_REPEAT.

Además de estas, tenemos la opción para restringir con:
GL_CLAMP,
GL_CLAMP_TO_BORDER
GL_CLAMP_TO_EDGE

Filtering

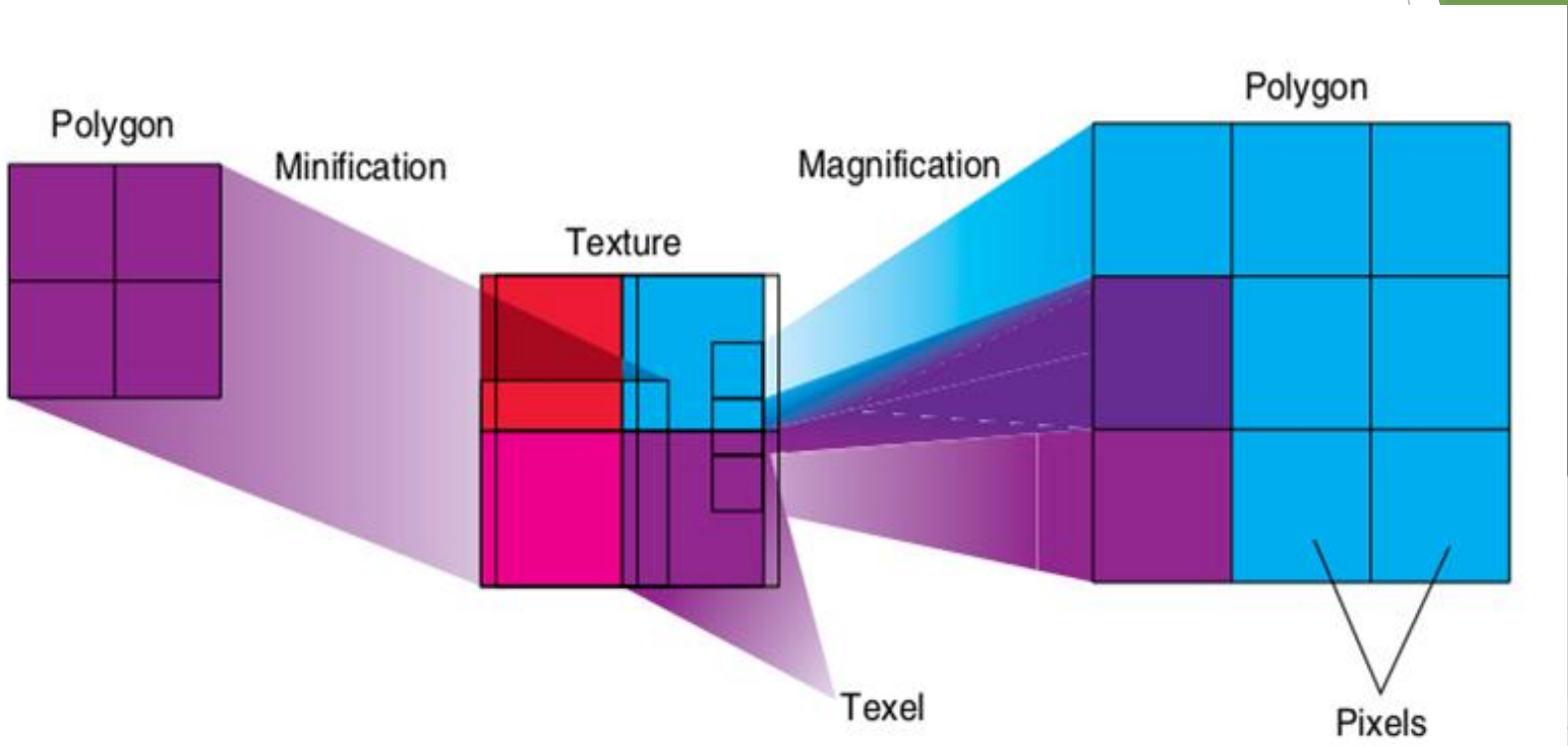
Las texturas usualmente son cuadrados o rectángulos

Después de ser mapeadas a un polígono o superficie y de ser transformadas a las coordenadas de pantalla, raramente los texeles individuales corresponden a pixeles individuales de la imagen final en pantalla.

Filtering

Dependiendo de las transformaciones usadas y el mapeo de textura aplicado, un solo pixel en pantalla puede corresponder a una pequeña porción de un texel o a una gran colección de texeles.

Filtering



Texturas

OpenGL

Filtering

En cualquier caso, no es claro cuales son los valores de los texeles que se deberían usar y como se promedian o interpolan.

Filtering

OpenGL permite especificar una o mas de las opciones de filtering para determinar estos cálculos

Estas opciones proveen diferentes ventajas y desventajas entre velocidad y calidad de imagen.

Filtering

También se puede especificar los métodos de filtering para la magnificación y minimización

Filtering

Si la textura necesita ser estirada en las direcciones 'x' y 'y', entonces se necesita la magnificación.

Si la textura necesita ser encogida en las direcciones 'x' y 'y', entonces s necesita minimización

Filtering

Si la textura necesita ser estirada en una dirección y encogida en la otra, OpenGL hace la elección entre la magnificación y la minimización, que en la mayoría de los casos nos da el mejor resultado posible.

Filtering

Nota: Lo mejor es evitar situaciones como la anterior usando coordenadas de textura que mapean sin distorsión

Filtering

Para poder usar estas opciones usamos la funcion glTexParameter*().

-glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER, GL_NEAREST);

-glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER, GL_NEAREST);

Filtering

El segundo parámetro es GL_TEXTURE_MAG_FILTER o GL_TEXTURE_MIN_FILTER, que no indican, claramente, si se utiliza método para magnificación o minimización

Filtering

El tercer argumento puede ser:
-GL_NEAREST o GL_LINEAR para la GL_TEXTURE_MAG_FILTER.
-GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_LINEAR.

Filtering

- Si se elige GL_NEAREST, el texel con las coordenadas mas cercanas al centro del pixel. Se utiliza tanto para la magnificacion y minimización

- Si se elige GL_LINEAR, se utiliza una matriz de 2x2 promedio de píxeles que se encuentran mas cerca al centro del pixel. Se utiliza tanto para la magnificacion y minimización.

Texturas

OpenGL

Filtering

Cuando las coordenadas de textura se encuentran cerca de la frontera de la textura, la matriz 2x2 de texeles mas cercanos puede incluir algunos que están fuera de la textura.

En estos casos, el valor del texel depende de cual modo wrapping esta en efecto y si se asigno un borde a la textura.

Filtering

GL_NEAREST requiere menos calculos que GL_LINEAR y por lo tanto se ejecuta mas rápido, pero GL_LINEAR provee mejores resultados.

Texturas

OpenGL

Filtering

Ejemplo:

Cambiar a linear los filtros en fieldAndSky.cpp y observar los cambios.

Texturas

OpenGL

Filtering

Finalmente tenemos que para los filtros la rutina glTexParameter*() queda:

glTexParameter*(GL_TEXTURE_2D, case, filter)

Filtering

Para poder hablar de las otras opciones de filtro es necesario definir lo que es un mipmap.

Filtering

Mipmaps

Los objetos texturizados se pueden ver a diferentes distancias del punto de visión.

En una escena dinámica, conforme se aleja el objeto texturizado del punto de visión, la textura debe reducir de tamaño junto con el de la imagen proyectada.

Filtering

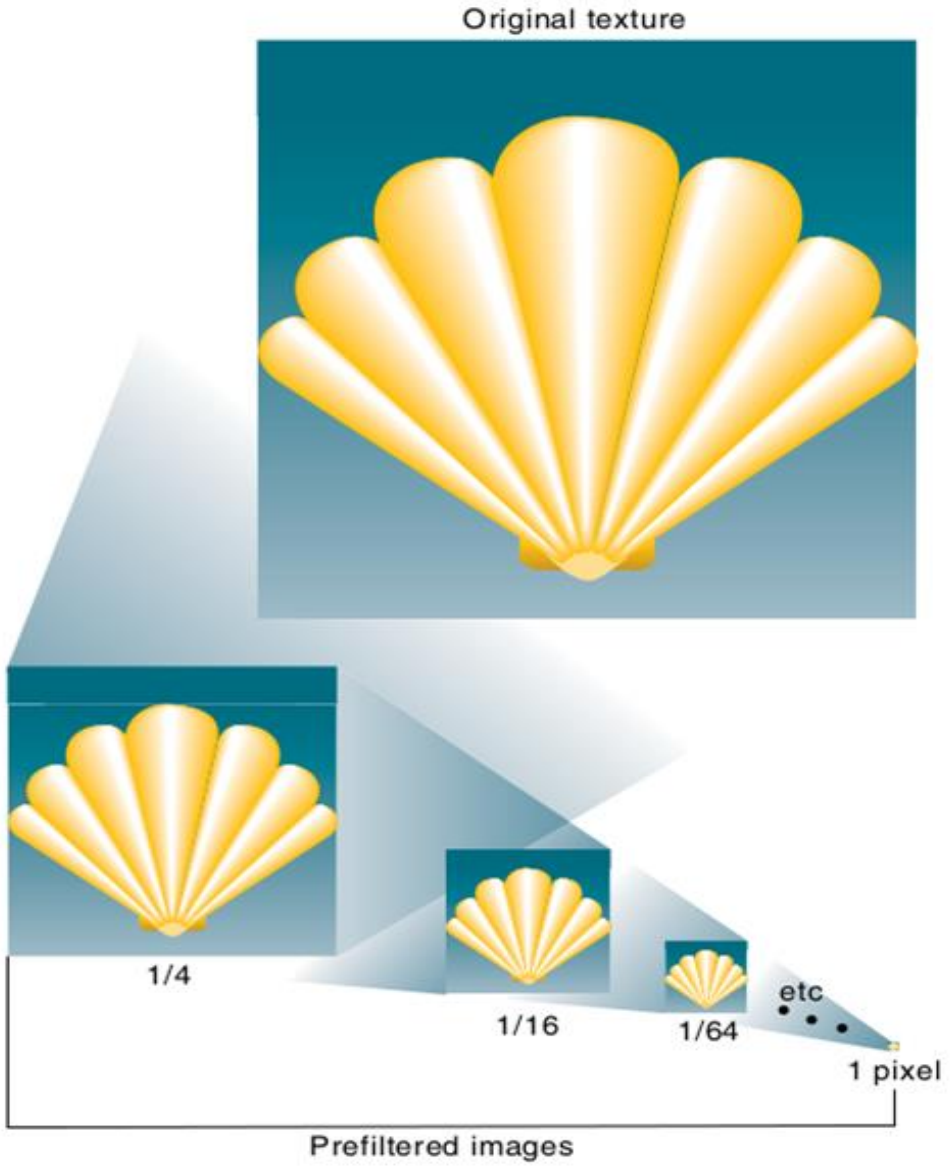
Mipmaps

Para lograr esto, OpenGL tiene que ajustar la textura a un tamaño apropiado para el mapeo en el objeto, sin agregar nada mas.

Filtering

Mipmaps

OpenGL puede especificar una serie de ajustes previos a la textura para el decremento de resolución Estos ajustes son llamados mipmaps.



Texturas

OpenGL

Filtering

Mipmaps

Cuando usamos mipmaping, OpenGL automáticamente determina cual textura se usa, basándose en el tamaño del objeto a mapear.

Filtering

Mipmaps

Con la magnificación, incluso si damos los mipmaps, solo el nivel base de la textura es usado.

Con la minimización podemos escoger un método de filtrado que el o los mipmaps mas apropiados.

Filtering

Para la minimización tenemos 4 opciones de filtrado adicionales con los mipmaps:

-GL_NEAREST_MIPMAP_NEAREST: Aplica el mipmap que este mas cercano a la resolución de la primitiva rasterizada y entonces usa GL_NEAREST en ese mipmap.

Filtering

-GL_LINEAR_MIPMAP_NEAREST: Aplica el mipmap que este mas cercano a la resolución de la primitiva rasterizada y entonces usa GL_LINEAR en ese mipmap.

-GL_NEAREST_MIPMAP_LINEAR: Encuentra los 2 mipmaps mas cercanos a la resolución de la primitiva rasterizada, entonces usa GL_NEAREST, dentro del mipmap para producir 2 conjuntos de valores de color y finalmente toma un promedio ponderado de los 2 conjuntos.

Filtering

-GL_LINEAR_MIPMAP_LINEAR: Encuentra los 2 mipmaps mas cercanos a la resolución de la primitiva rasterizada, entonces usa GL_LINEAR, dentro del mipmap para producir 2 conjuntos de valores de color y finalmente toma un promedio ponderado de los 2 conjuntos.

Filtering

Para poder usar mipmaps podemos usar la rutina gluBuildMipmaps() de la biblioteca de GLU para generarlos automaticamente.

-GLint gluBuild2DMipmap(GLenum target, GLint internalFormat,
GLsizei width, GLsizei height,
GLenum format, GLenum type,
const void * data).

Filtering

-target especifica el tipo de textura

-internalFormat indica el formato en el que se almacena la textura.

-width, height indica las dimensiones de la textura.

-format especifica el formato de los píxeles

-type especifica el tipo de dato para data

-data es el apuntador de la imagen.

Texturas

OpenGL

Filtering

Observemos en el código fieldAndSkyFiltered.cpp que en el caso en el que usamos mipmaps en lugar de usar glTexImage2D() se reemplaza por gluBuild2DMipmaps() para generar todos los mipmaps de las texturas bases.

```
GluBuild2DMipmaps( GL_TEXTURE_2D,
                   GL_RGB,image[0]->sizeX,
                          image[0]->sizeY, GL_RGB,
                   GL_UNSIGNED_BYTE, image[0]->data);
```

Filtering

En el código compareFilters.cpp podemos ver la comparación de la calidad de las texturas usando los filtros de minimización y magnificacion.

Filtering

Existe varias formas de crear un mipmap usando las caracteristicas de OpenGL.
En las ultimas versiones podemos usar el método glGenerateMipmap() el cual nos da la pila de mipmaps para la textura actual.

Filtering

-int glGenerateMipmap(GLenum target)
Genera un conjunto completo de mipmaps para la textura asociada target.
Los niveles de mipmaps construidos son controlados por GL_TEXTURE_BASE_LEVEL y GL_TEXTURE_MAX_LEVEL

Filtering

Si no se tiene acceso a versiones mas recientes de OpeGL se pueden generar mipmaps con OpenGL usando glTexParameter*() para asignar GL_GENERATE_MIPMAP a GL_TRUE, entonces cualquier cambio en los texeles de un mipmap BASE_LEVEL causa automáticamente que todas las texturas en los niveles desde BASE_LEVEL + 1 hasta MAX_LEVEL se recalculen y remplacen .

Filtering

Si se tiene una versión mas antigua de OpenGL se necesita construir la pila manualmente.

Filtering

Para asignar la base y el máximo nivel de mipmap, usamos glTexParameter*() con el primer argumento GL_TEXTURE_1D, 2D, 3D, CUBE_MAP, dependiendo de la textura.

Filtering

El segundo argumento es uno de los siguientes:

- GL_TEXTURE_BASE_LEVEL nivel para la textura de mayor resolución.

- GL_TEXTURE_MAX_LEVEL nivel para la textura de menor resolución.

Texturas iluminadas

OpenGL nos ofrece diferentes opciones para combinar las luces, color y texturas.

Una opción es con la funcion glTextEnv*().

```
GlTexEnvf( GL_TEXTURE_ENV,
           GL_TEXTURE_ENV_MODE,
            parameter)
```

Texturas iluminadas

Si parameter es GL_REPLACE, entonces los colores usados para renderear cada primitiva son derivadas solamente de la textura usada.

El color del material de la primitiva así como las fuentes de luz en el ambiente son ignoradas.

Texturas iluminadas

La forma mas común para combinar color y luz con la textura es asignado al parameter de glTexEnv()* el valor GL_MODULATE.

Texturas iluminadas

En caso de usar GL_MODULATE se realiza los siguiente:

1.- Calcula los valores de RGB en los vértices de una primitiva usando la iluminación de OpenGL e interpolando estos valores en el interior, se asume que smooth esta activado, para determinar los valores RGB en cada uno de los píxeles.

Texturas iluminadas

2.- Usa la textura para obtener los valores RGB de la textura en cada uno de los píxeles de la primitiva.

3.- Determina los valores finales de RGB en cada píxel como el producto de los valores correspondientes de los 2 pasos anteriores.

Texturas iluminadas

En conclusión, OpenGL calcula por separado los valores RGB para el color y luz como si no hubiera textura y los valores RGB para la textura como si no hubiera color y luz, finalmente escala uno con el otro.