

## 10. EL MAPEADO DE TEXTURAS

Hasta ahora las primitivas se han dibujado en OpenGL con un solo color o interpolando varios colores entre los vértices de una primitiva. OpenGL dispone de funciones específicas para el mapeado de texturas (pegar imágenes en la superficie de las primitivas dibujadas), añadiendo realismo a la escena. En este capítulo se explica el proceso de mapeado de texturas a través de un sencillo ejemplo, para posteriormente emplear las texturas en la aplicación tecnunLogo.

### 10.1 CARGANDO Y PEGANDO UNA TEXTURA

En este capítulo se va a utilizar la función **LoadDIBitmap** (de Michael Sweet, *OpenGL SuperBible*) para cargar archivos gráficos de tipo bmp. Para esto se deben incluir al directorio del proyecto los archivos *bitmap.c* y *bitmap.h* y añadirlos al proyecto (Project / Add to Project / Files...). También es necesario incluir el archivo “escudo.bmp”. Los archivos se encuentran en la dirección:

<http://www.tecnun.es/asignaturas/grafcomp/practicas/textures/escudo.bmp>

<http://www.tecnun.es/asignaturas/grafcomp/practicas/textures/bitmap.c>

<http://www.tecnun.es/asignaturas/grafcomp/practicas/textures/bitmap.h>

El código del ejemplo es el siguiente:

```
#include <GL/glut.h>
#include "bitmap.h"

BITMAPINFO *TexInfo; /* Texture bitmap information */
GLubyte *TexBits; /* Texture bitmap pixel bits */

void display(void) {

    glClearColor (1.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    TexBits = LoadDIBitmap("escudo.bmp", &TexInfo);

    glTexImage2D (GL_TEXTURE_2D, 0, 3, TexInfo->bmiHeader.biWidth,
                  TexInfo->bmiHeader.biHeight, 0, GL_BGR_EXT,
                  GL_UNSIGNED_BYTE, TexBits);

    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

    glColor3f(1.0, 1.0, 1.0);

    //se activa el mapeado de texturas
    glEnable (GL_TEXTURE_2D);

    glBegin (GL_POLYGON);
    glTexCoord2f(0.0, 0.0);
    glVertex2f(-1.0, -1.0);
    glTexCoord2f(1.0, 0.0);
    glVertex2f(1.0, -1.0);
```

```

    glVertex2f(1.0, 1.0);
    glVertex2f(1.0, 1.0);
    glVertex2f(0.0, 1.0);
    glVertex2f(-1.0, 1.0);
    glEnd();

    glDisable(GL_TEXTURE_2D);
    glutSwapBuffers();
}

void reshape(int width, int height) {
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (GLfloat)height / (GLfloat)width, 1.0, 128.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 1.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("tecnunLogo");

    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}

```

Como puede observarse, la función `main()` y `reshape()` son muy similares a las ya vistas en prácticas anteriores. Para entender el proceso de mapeado de texturas se va a explicar únicamente la función `display()`.

Las texturas en OpenGL pueden ser 1D, 2D o 3D. Las 1D tienen anchura pero no altura; las 2D son imágenes que tienen una anchura y una altura de más de 1 píxel, y son generalmente cargadas a partir de un archivo `.bmp` (aunque puede ser en principio cualquier formato). En este capítulo no se hablará de las texturas 3D (volumen). Un aspecto muy importante a tener en cuenta es que en OpenGL las dimensiones de las imágenes deben ser potencia de 2 (2, 4, 8, 16, 32...)

Antes de definir la textura, se carga una imagen a partir de un archivo (`.bmp` en este caso). Para esto es necesario incluir dos punteros, `TexInfo` y `TexBits`, para almacenar la información de la imagen (header) y los bits que componen la imagen respectivamente. Se carga la imagen con la función **`LoadDIBitmap()`** y se le pasa como argumentos el nombre de la imagen (“escudo.bmp” en este caso) y el puntero que guardará la información de dicha imagen (header).

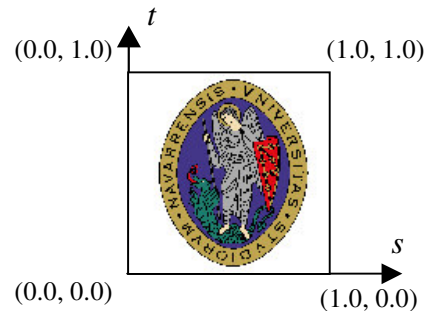
Para definir las texturas, OpenGL pone a disposición las funciones **`glTexImage1D`** y **`glTexImage2D`** (la única diferencia entre ambas es que en la segunda se define, además, la altura de la imagen). La función **`glTexImage2D`** tiene 9 argumentos:

```
void glTexImage2D(GLenum target, GLint level, GLint components, GLsizei width,
                 GLsizei height, GLint border, GLenum format, GLenum type,
                 const GLvoid *pixels)
```

El *target* indica el tipo de textura que va a definirse, en este caso `GL_TEXTURE_2D`. *Level* el nivel de detalle de la imagen, generalmente se utiliza el valor 0. *Components* el número de colores que han sido utilizados en la definición de cada píxel de la imagen, en este caso 3. *Width* y *height* son las dimensiones de la imagen, siempre potencia de 2, y se accede a esta información por medio de la variable `TexInfo`. *Border* indica el número de píxeles que debe tener el borde de la textura, y toma valores de 0, 1 o 2. *Format* el tipo de colores que se espera tenga la textura. *Type* el tipo de datos en que va a ser pasada la información de la imagen cargada, y *pixels* es la información en bits de esa imagen, que en este caso está almacenada en la variable `TexBits`.

Una vez cargada la imagen y definida la textura, se define la manera de cómo será aplicada la textura a cada píxel de la superficie que queremos texturizar por medio de la función `glTexParameterf()`.

La textura tiene 2 coordenadas, la *s* para el eje horizontal y la *t* para el eje vertical, y toma valores de 0.0 a 1.0. Para la coordenada *s* 0.0 representa el extremo izquierdo de la textura y 1.0 el extremo derecho. Para la *t* el 0.0 representa el extremo inferior y el 1.0 el superior.



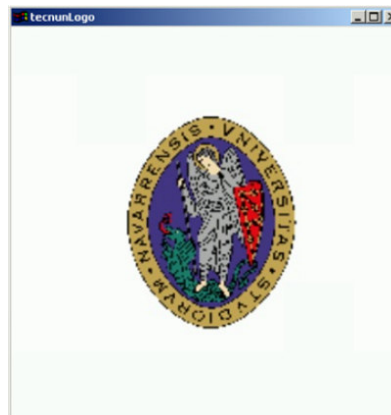
**Texture Wrap.** Cuando se pega la textura a una superficie debe indicarse la correspondencia entre las coordenadas de la textura los vértices de dicha superficie. Si las coordenadas que hacen referencia a la textura se salen del rango 0-1, la textura se repite (`GL_REPEAT`) o se estira (`GL_CLAMP`) sobre la superficie según se defina en el `TEXTURE_WRAP`. En este ejemplo se define que, en ambas direcciones *s* y *t*, la textura se repita:

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

**Texture Filters.** Cuando la imagen de la textura no se adecua con el tamaño de la superficie que se quiere cubrir, OpenGL utiliza unos filtros de textura para interpolar entre los píxeles de la imagen y adecuarla a la superficie. Cuando la superficie sea menor que la imagen OpenGL utilizará un *minification filter* (`GL_TEXTURE_MIN_FILTER`) y cuando la superficie sea mayor que la imagen, utilizará un *magnification filter* (`GL_TEXTURE_MAG_FILTER`). OpenGL pone a disposición 6 tipos de filtros (ver tutoriales). En este caso se ha optado por una interpolación lineal:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

Una vez se tiene definida completamente la textura, se activa el mapeado de texturas por medio de la función **glEnable(GL\_TEXTURE\_2D)** y se dibuja la escena, indicando la correspondencia entre las coordenadas de la textura y las coordenadas de los vértices de la superficie. Para hacer referencia a las coordenadas de la textura se utiliza la función **glTexCoord2f()** (o **glTexCoord3f()**, según sea el caso).



## 10.2 RESUMEN

El mapeado de texturas puede resumirse en cuatro pasos:

1. Se carga una imagen y se define como textura
2. Se indica cómo va a aplicarse la textura a cada píxel de la superficie
3. Se activa el mapeado de texturas
4. Se dibuja la escena, indicando la correspondencia entre las coordenadas de la textura y los vértices de la superficie.

## 10.3 CARGANDO Y PEGANDO UNA TEXTURA EN TECNUNLOGO

Para cargar una imagen y pegar una textura en `tecnunLogo` se van a modificar las funciones **main()** y **drawTurtle()** para no alterar el funcionamiento de la función **display()** de la aplicación.

Se deben añadir los ficheros *bitmap.c* y *bitmap.h* al directorio del proyecto y añadirlos al proyecto (Project/ Add to Project/ Files...).

Incluir en `tecnunLogo` el *include* de `bitmap.h`:

```
#include "bitmap.h"
```

Incluir en las declaraciones, los punteros para almacenar la información (header) de la imagen y los bits que componen la imagen:

```
BITMAPINFO *TexInfo;
GLubyte *TexBits;
```

Dentro de la función **main()**, se carga la imagen que se utilizará posteriormente y se indica cómo deberá aplicarse a la superficie:

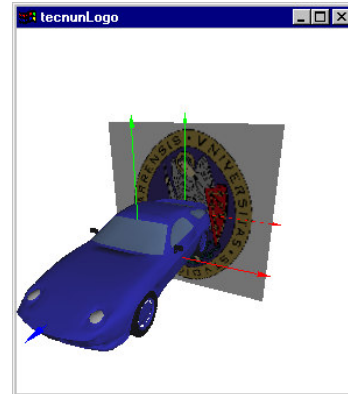
```
TexBits = LoadDIBitmap("escudo.bmp", &TexInfo);

glTexImage2D(GL_TEXTURE_2D, 0, 3, TexInfo->bmiHeader.biWidth,
             TexInfo->bmiHeader.biHeight, 0, GL_BGR_EXT,
             GL_UNSIGNED_BYTE, TexBits);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

En la función **drawTurtle()** comentar el bucle que genera el dibujo de la tortuga (desde **glBegin()** hasta **glEnd()**) y sustituir por el código que activa el mapeado de texturas y dibuja un rectángulo, indicando la correspondencia entre las coordenadas de la textura y los vértices de dicho rectángulo:

```
glEnable(GL_TEXTURE_2D);  
glBegin(GL_POLYGON);  
glTexCoord2f(0.0, 0.0);  
glVertex2f(-1.0, -1.0);  
glTexCoord2f(1.0, 0.0);  
glVertex2f(1.0, -1.0);  
glTexCoord2f(1.0, 1.0);  
glVertex2f(1.0, 1.0);  
glTexCoord2f(0.0, 1.0);  
glVertex2f(-1.0, 1.0);  
glEnd();  
glDisable(GL_TEXTURE_2D);
```



**Figura 1.** tecnunLogo con Imagen

Para que se dibuje la tortuga será necesario activar una tortuga distinta de la 0, por ejemplo con el comando “st 1”. En la figura se aprecia el resultado.

#### 10.4 TRABAJO PROPUESTO

*Añadir una textura al rastro que deja la tortuga al hacer un recorrido. El rastro deberá ser un polígono o una serie de polígonos para facilitar la definición de las coordenadas de textura.*