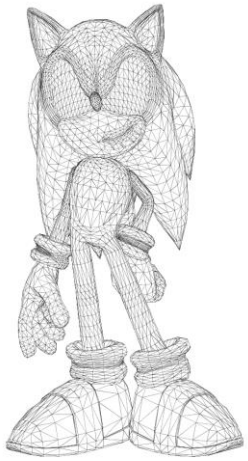
Programación 3D Tema 2 - OpenGL



Master en Programación de Videojuegos

Javier Alegre Landáburu
javier.alegre@u-tad.com

Coordenadas de textura

- Cada vértice, además de su posición, debe contener información sobre qué punto de la textura debe aplicarse al mismo.

- Estas coordenadas de textura se llaman coordenadas UV (o ST), y en OpenGL siguen el sistema de la imagen de la derecha.

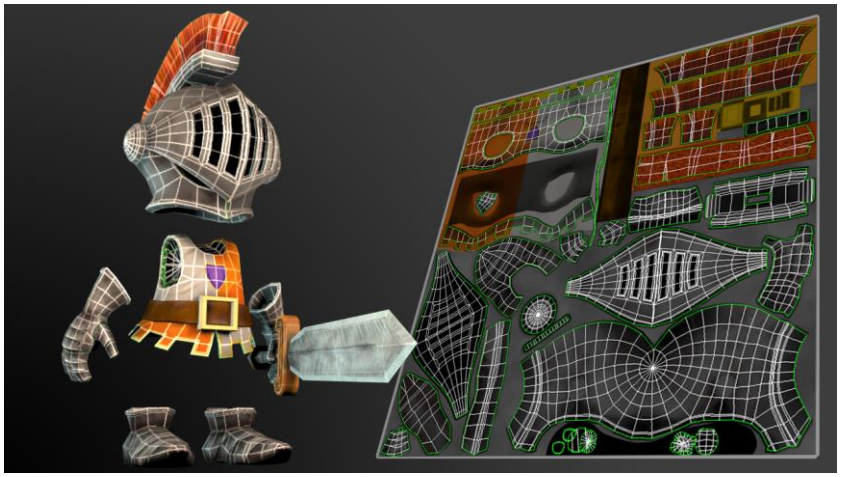
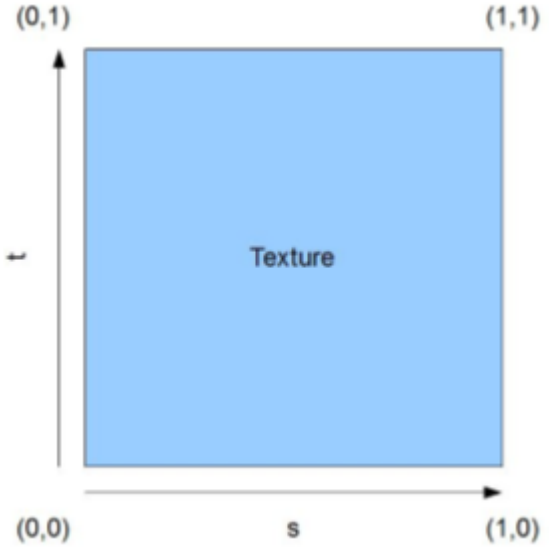


Imagen obtenida en
https://www.pluralsight.com/courses/3ds-max-uv-mapping-fundamentals



Coordenadas de textura

- Dependiendo del tamaño del objeto al que se aplica la textura (y de su distancia al observador en una proyección con perspectiva), será necesario escalar la textura a un tamaño menor.

- OpenGL se encarga de hacer esto automáticamente, y podemos definir distintos tipos de filtrado para el escalado.

Coordenadas de textura

- Es posible acelerar el proceso de escalado de la textura si utilizamos mipmapping:
  - Esta técnica consiste en generar diversas imágenes, cada una de la mitad de tamaño que la anterior.
  - A la hora de dibujar la textura, se escoge aquella que se aproxima más al tamaño final.

Carga de ficheros de imagen

- Las texturas se generan a partir de un array de bytes que contienen los componentes RGBA de cada píxel de la textura, en el rango [0, 255].
- Las tarjetas antiguas tenían como limitación que las texturas tengan un ancho y un alto que sean potencia de 2 (2x2, 16x16, 128x256...).
- Estos arrays de bytes los obtendremos a partir de ficheros de imagen.
- Existen diversos formatos de imagen: BMP, PNG, JPG…
- Algunos de ellos, como el BMP, son muy sencillos de cargar directamente. Otros no guardan la información que queremos directamente, sino comprimida.

Carga de ficheros de imagen

- Para obtener el array de bytes que queremos, podemos utilizar una librería de carga de imágenes.
- En nuestro motor, emplearemos STB Image, librería de dominio público.
- Utilizaremos dos únicas funciones de dicha librería: **stbi_load** y **stbi_image_free**.
- Debemos incluir el archivo de cabecera "stb_image.h".
- En un sólo fichero de código fuente, antes de incluir el archivo, definiremos la macro STB_IMAGE_IMPLEMENTATION.

Carga de ficheros de imagen

unsigned char* stbi_load(const char* filename, int* x, int* y, int* comp, int req_comp)

Devuelve un array de bytes con los datos de una imagen. Parámetros:

- **filename**: Nombre del fichero.
- **x, y**: Punteros donde se guardará el ancho y alto de la imagen.
- **comp**: Puntero donde se guardará el número de componentes del buffer (podemos poner **nullptr** si no nos interesa esta información).
- **req_comp**: Podemos forzar el número de componentes aquí (indicaremos 4, RGBA).

- Si no se ha podido cargar la imagen, devuelve **nullptr**.

Carga de ficheros de imagen

void stbi_image_free(void* buffer)

- Elimina un buffer generado con la función anterior (es lo mismo que utilizar directamente la función free de C).

Generación de texturas con OpenGL

Una vez que tenemos el array con los datos de la imagen, debemos generar una textura con OpenGL.

El proceso consta de cinco pasos:

1. Generación.
2. Activación.
3. Establecimiento de parámetros.
4. Carga de la imagen.
5. Generación de mipmaps.

Generación de texturas con OpenGL

Paso 1: Generación

Generamos la textura con la función **glGenTextures.** Debemos indicarle el número de texturas a generar y un puntero donde guardará el identificador de cada una.

Ejemplo:

```
GLuint texId; glGenTextures(1, &texId);
```

Generación de texturas con OpenGL

Paso 2: Activación

Debemos convertir la textura generada en la activa con la función **glBindTexture** (todas las operaciones de textura trabajarán sobre la activa).

Ejemplo:

```
glBindTexture(GL_TEXTURE_2D, texId);
```

Generación de texturas con OpenGL

Paso 3: Establecimiento de parámetros

Establecemos los parámetros de la textura con **glTexParameteri**, que recibe como parámetros:

glTexParameteri(**Tipo**, **Parámetro**, **Valor**);

Tipo: Debe ser GL_TEXTURE_2D o GL_TEXTURE_CUBE_MAP.

Parámetro: El parámetro a establecer.

Los parámetros que trataremos en la asignatura son:
GL_TEXTURE_MIN_FILTER, GL_TEXTURE_MAG_FILTER, GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T.

Valor: Valor del parámetro (Continua en la sig. trasparencia).

Generación de texturas con OpenGL

Valor: Valor del parámetro.

GL_TEXTURE_MIN_FILTER: Cuando una textura se va a dibujar más pequeña de su tamaño original, establece el tipo de filtrado.

GL_TEXTURE_MAG_FILTER: Igual que la anterior, pero se utiliza cuando la textura se va a dibujar más grande.

Valores que toman (**GL_TEXTURE_MAG_FILTER** sólo puede tomar los dos primeros):

- **GL_NEAREST**: Sin interpolación.
- **GL_LINEAR**: Con interpolación.
- **GL_NEAREST_MIPMAP_NEAREST**: Escoge el mipmap más próximo al tamaño a dibujar, y lo aplica sin interpolación.
- **GL_LINEAR_MIPMAP_NEAREST**: Interpola los mipmaps más próximos al tamaño a dibujar, y aplica el resultado sin interpolación.
- **GL_NEAREST_MIPMAP_LINEAR**: Escoge el mipmap más próximo al tamaño a dibujar, y lo aplica con interpolación (bilinear).
- **GL_LINEAR_MIPMAP_LINEAR**: Interpola los mipmaps más próximo al tamaño a dibujar, y aplica el resultado con interpolación (trilinear).

Generación de texturas con OpenGL

Valor: Valor del parámetro.

GL_TEXTURE_WRAP_S: Cuando la coordenada de mapeado horizontal de la textura está fuera del rango [0, 1], indica la forma en que se mapea la textura.

GL_TEXTURE_WRAP_T: Cuando la coordenada de mapeado vertical de la textura está fuera del rango [0, 1], indica la forma en que se mapea la textura.

Valores que toma:

GL_REPEAT: La textura se dibuja repetida a lo largo de la malla (valor por defecto).

GL_MIRRORED_REPEAT: La textura se dibuja repetida a lo largo de la malla, pero invirtiendo en el eje correspondiente cada vez que se repite.

GL_CLAMP_TO_EDGE: Repite la coordenada de textura 0 si el valor es negativo, o la coordenada de textura 1 si el valor es mayor que 1.

Generación de texturas con OpenGL

Paso 4: Carga de la imagen

Cargamos el buffer en la textura con glTexImage2D, que recibe como parámetros:

glTexImage2D (GLenum target, GLint level, GLint internalformat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const void *pixels);

Tipo: Debe ser GL_TEXTURE_2D.

Nivel: Nivel de mipmap (utilizaremos 0).

Formato interno: Formato de la textura (utilizaremos GL_RGBA).

Ancho, alto: Tamaño de la imagen.

Borde: Debe ser 0.

Formato: Formato del buffer (utilizaremos GL_RGBA).

Tipo: Tipo de datos en el buffer (utilizaremos GL_UNSIGNED_BYTE).

Buffer: Puntero al buffer de datos.

Generación de texturas con OpenGL

Paso 5: Generación de mipmaps

Los mipmaps se pueden generar automáticamente si estamos usando filtrado bilinear o trilinear una vez se han definido los datos de la textura utilizando la función **glGenerateMipmap(GL_TEXTURE_2D).**

Una vez generada la textura, podemos eliminar el buffer de **STB_Image** con **stbi_image_free**, ya que OpenGL ha creado una copia de los píxeles en memoria de vídeo.

Generación de texturas con OpenGL

- Para dibujar las primitivas utilizando una textura, utilizaremos glBindTexture(GL_TEXTURE_2D, texId). A esto se le llama enlazar la textura.
- OpenGL soporta varias unidades de textura, para permitir que un mismo triángulo tenga varias texturas (volveremos a esto al final de la asignatura).
-
- Con la función glUniform1i, escribiremos en una variable uniform de tipo sampler2D del fragment shader la unidad en la que hemos enlazado la textura (por defecto 0).
- El tipo sampler2D nos permite acceder a los datos enlazados a una unidad de textura.
-
- También debemos escribir en una variable attribute de tipo vec2 del vertex shader las coordenadas de mapeado del vértice.
- Como la textura la aplicaremos en el fragment shader, debemos pasar esta información en una variable varying.
- La función texture2D nos permite extraer el color de un determinado píxel de la textura.

Generación de texturas con OpenGL

Código del vertex shader:

```
uniform mat4 MVP;
attribute vec3 vpos;
attribute vec2 vtex;
varying vec2 ftex;
void main()
{
gl_Position = MVP * vec4(vpos, 1);
ftex = vtex;
}
```

Generación de texturas con OpenGL

Código del fragment shader:

```
uniform sampler2D texSampler;
varying vec2 ftex;
void main()
{
        gl_FragColor = texture2D( texSampler, ftex);
}
```