Texturas con WebGL

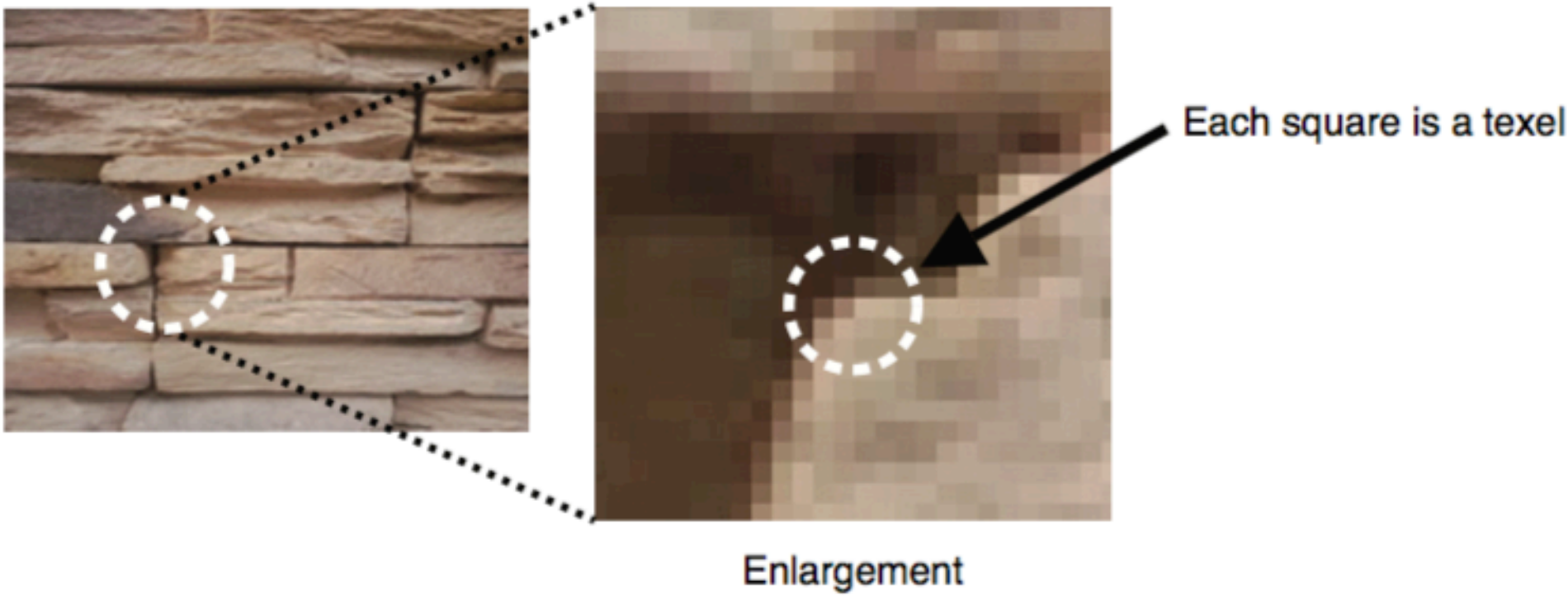
Katia Leal Algara
Web: http://gsyc.urjc.es/~katia/
Email: katia.leal@urjc.es

Dept. Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación (GSyC)
Escuela Superior De Ingeniería De Telecomunicación (ETSIT)
Universidad Rey Juan Carlos (URJC)

Proceso de texturización

- El proceso de texturización consiste en asignar el color de los píxeles de una imagen (textura) a los fragmentos generados en la fase de **rasterización**.
- Los píxeles que forman la textura se denominan **texels** (elementos de textura) y cada uno codifica la información de su color en el formato RGB o RGBA.
- Los texels son la unidad mínima de una textura aplicada a una superficie.

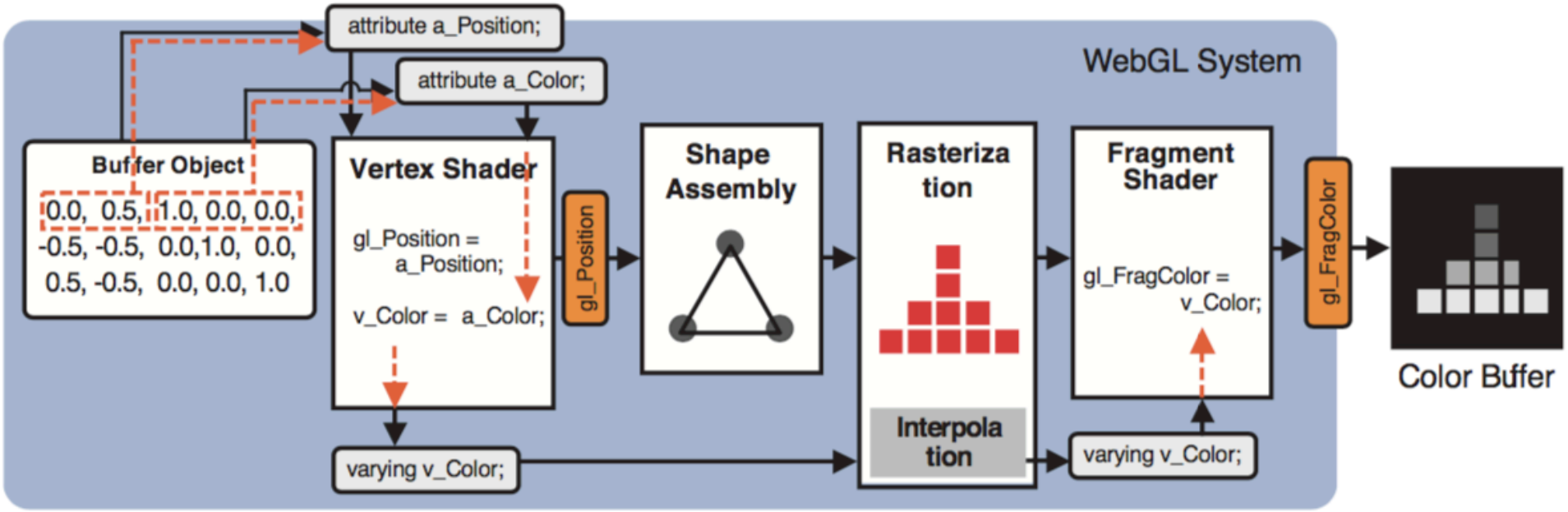
Proceso de texturización



Rasterización

- **Rasterización:** proceso por el cual una imagen descrita en un *formato gráfico vectorial* se convierte en un conjunto de *píxeles* o puntos para ser enviados a un medio de salida digital: pantalla, impresora electrónica o una Imagen de mapa de bits (bitmap).
- Este procedimiento se suele usar en momentos muy concretos:
  - Con imágenes de gran complejidad (con muchos objetos independientes, degradados, capas, etc.). Como al crear un mapa de bits se elimina toda información de los objetos vectoriales, se deben efectuar copias de seguridad del archivo vectorial antes de ser rasterizado o bien esperar a que la parte de la imagen que se va a rasterizar sea ya definitiva.
  - Cuando se van a aplicar filtros a la imagen resultante, cosa que no se efectúa con los objetos iniciales.

Rasterización



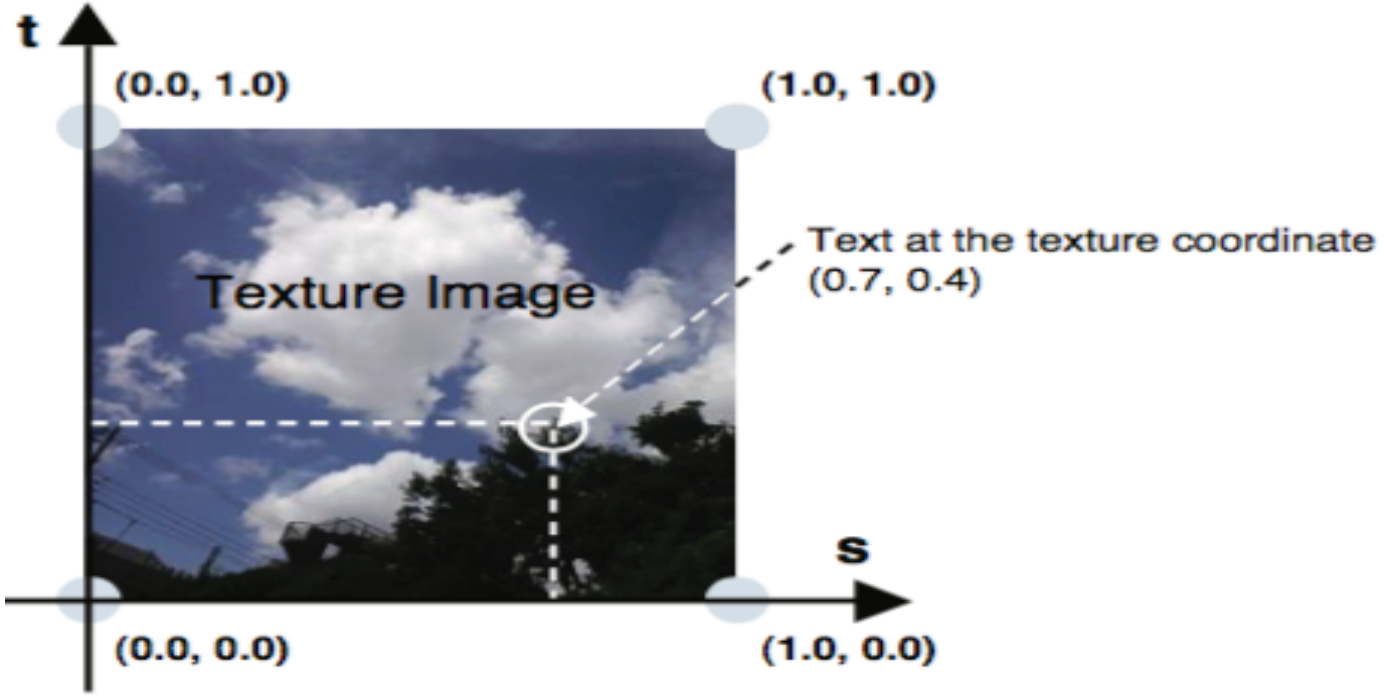
Mapeado de texturas

- En WebGL el mapeado de texturas incluye los siguientes pasos:

1. Preparar la imagen para mapearla en la forma geométrica.
2. Especificar el método de mapeo de imágenes para la forma geométrica.
3. Cargar la imagen de textura y configurarla para usarla en WebGL.
4. Extraer los texels de la imagen en el fragment shader y hacer corresponder con el fragmento correspondiente.

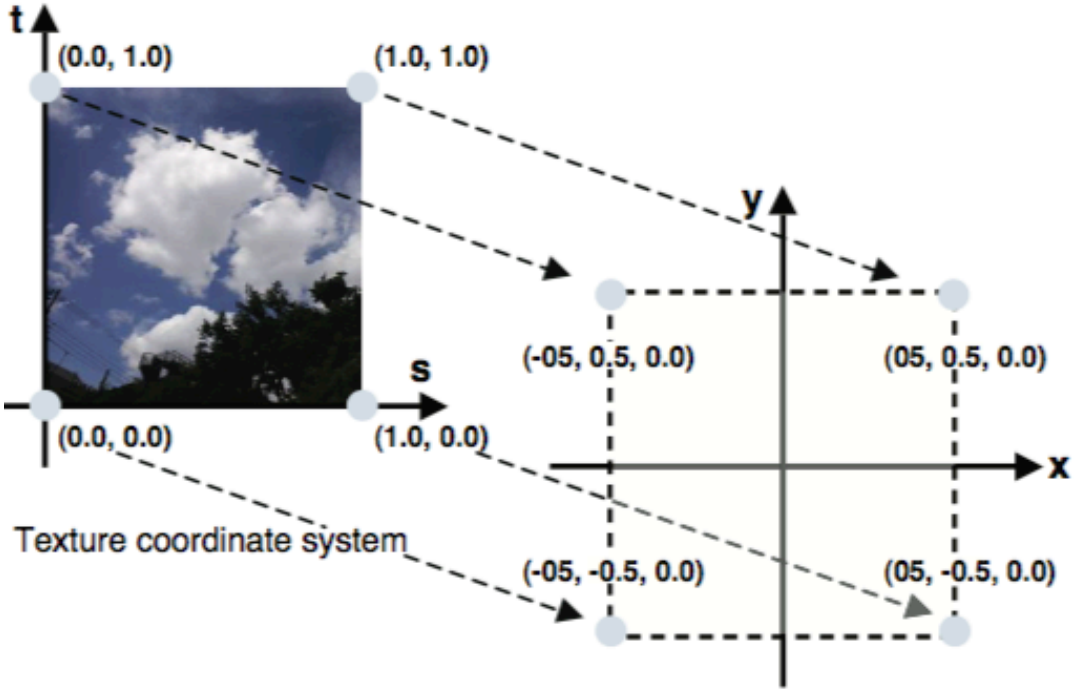
Coordenadas de textura

- El sistema de coordenadas de textura utilizado por WebGL es de dos dimensiones.
- Para no confundir las coordenadas de textura con los ejes **x** e **y**, WebGL utiliza las coordenadas **s** y **t**.



Pegar la textura en la forma

- En WebGL puedes especificar cómo se pegará la imagen de textura definiendo la correspondencia entre las coordenadas de textura y las coordenadas vertex de la forma geométrica.



The resulting image produced by WebGL

Pasos

1. Recibir las coordenadas de la textura en el vertex shader y pasárselas al fragment shader.
2. Pegar la forma geométrica en la forma dentro del fragment shader.
3. Inicializar las coordenadas de la textura.
4. Preparar la imagen de textura para su carga y solicitar al navegador que la lea.
5. Configurar la textura cargada de tal forma que pueda ser utilizada por WebGL.

Inicialización y carga de texturas

- **createTexture()** creates and initializes a WebGLTexture object.

  Syntax

  gl.createTexture();

  Parameters

  None.

  Return value

  A WebGLTexture object to which images can be bound to.

Inicialización y carga de texturas

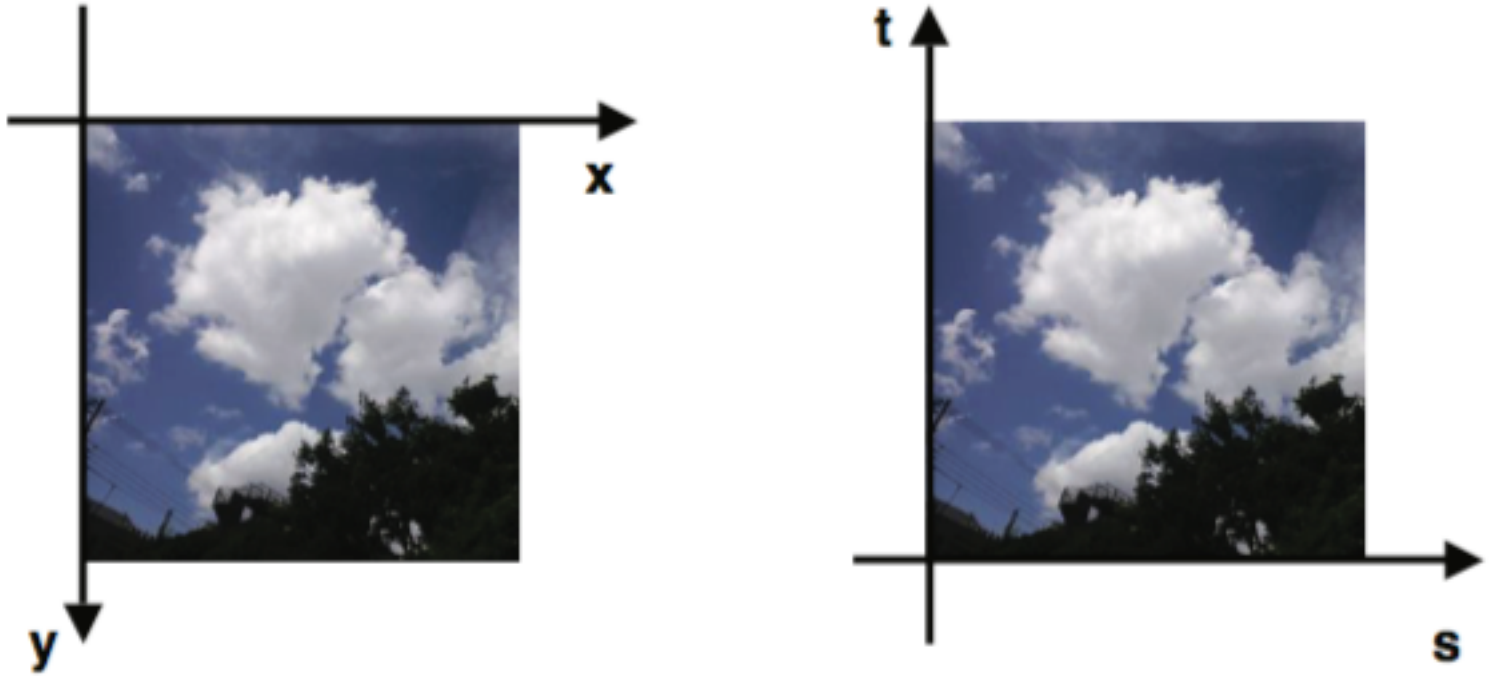
```
var image = new Image(); // Create an Image object
…
image.onload = function(){ loadTexture(gl, n,
                texture, u_Sampler, image); };
```

- La carga de la imagen se realiza de forma **asíncrona**.
  - La petición se realiza al navegador.
  - No se bloquea la ejecución del proceso.
- Se debe especificar una URL, de lo contrario, si se trata de ficheros locales, se tendrán problemas:
  - Cross-Origin Resource Sharing (**CORS**).
  - python -m SimpleHTTPServer (levantar servidor local).

Inicialización y carga de texturas

```
gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);
```

- Voltear el eje Y de la imagen antes de usarla como una textura.



Activar una unidad de textura

```
gl.activeTexture(gl.TEXTURE0);
```

- WebGL soporta múltiples imágenes de textura por medio de un mecanismo denominado *texture unit*.
- La texture unit utiliza un número de unidad para cada textura.
- Así, aunque sólo se use una textura, se debe indicar un número de unidad.
- El máximo número de unidades de textura depende de la implementación (hardware y WebGL):
  - gl.getParameter(gl.MAX_COMBINED_TEXTURE_IMAGE_UNITS);

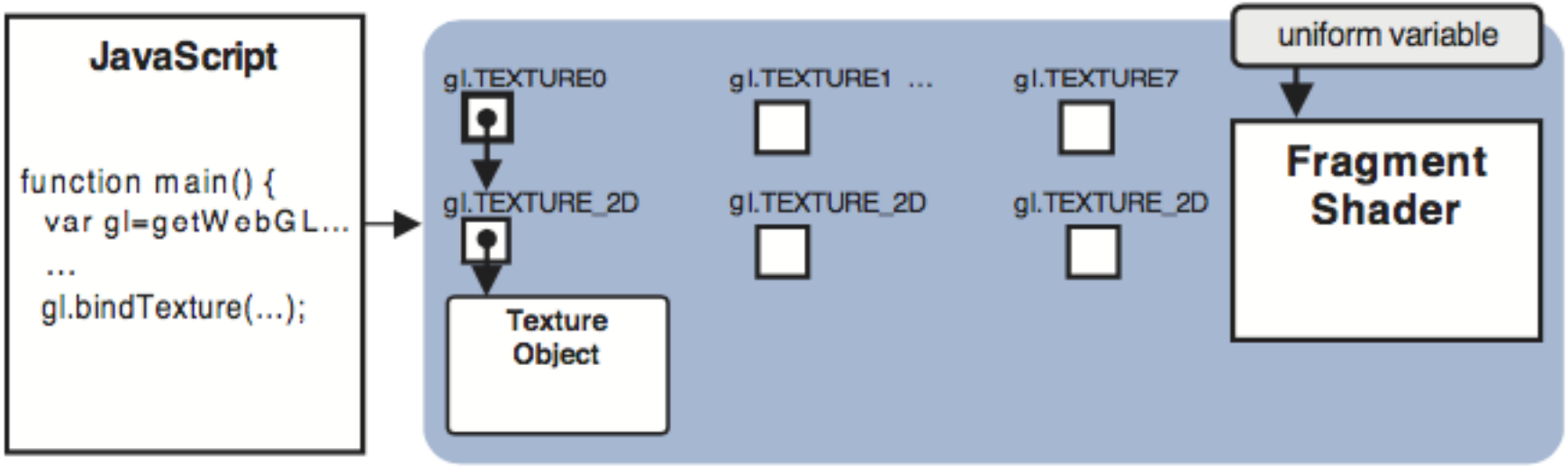
Enlazar un objeto textura con un objetivo

```
gl.bindTexture(gl.TEXTURE_2D, texture);
```

- Tipos de texturas:

  gl.TEXTURE_2D ➔ Two-dimensional texture.

  gl.TEXTURE_CUBE_MAP ➔ texture Cube map texture.



Establecer los parámetros del objeto textura

```
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
```

gl.texParameteri(target, pname, param): Set the value specified by *param* to the texture parameter specified by *pname* in the texture object bound to *target*.

Parameters:

- target: specifies gl.TEXTURE_2D or gl.TEXTURE_CUBE_MAP.
- pname: specifies the name of the texture parameter.
- param: specifies the value set to the texture parameter *pname.*

Return value: none.

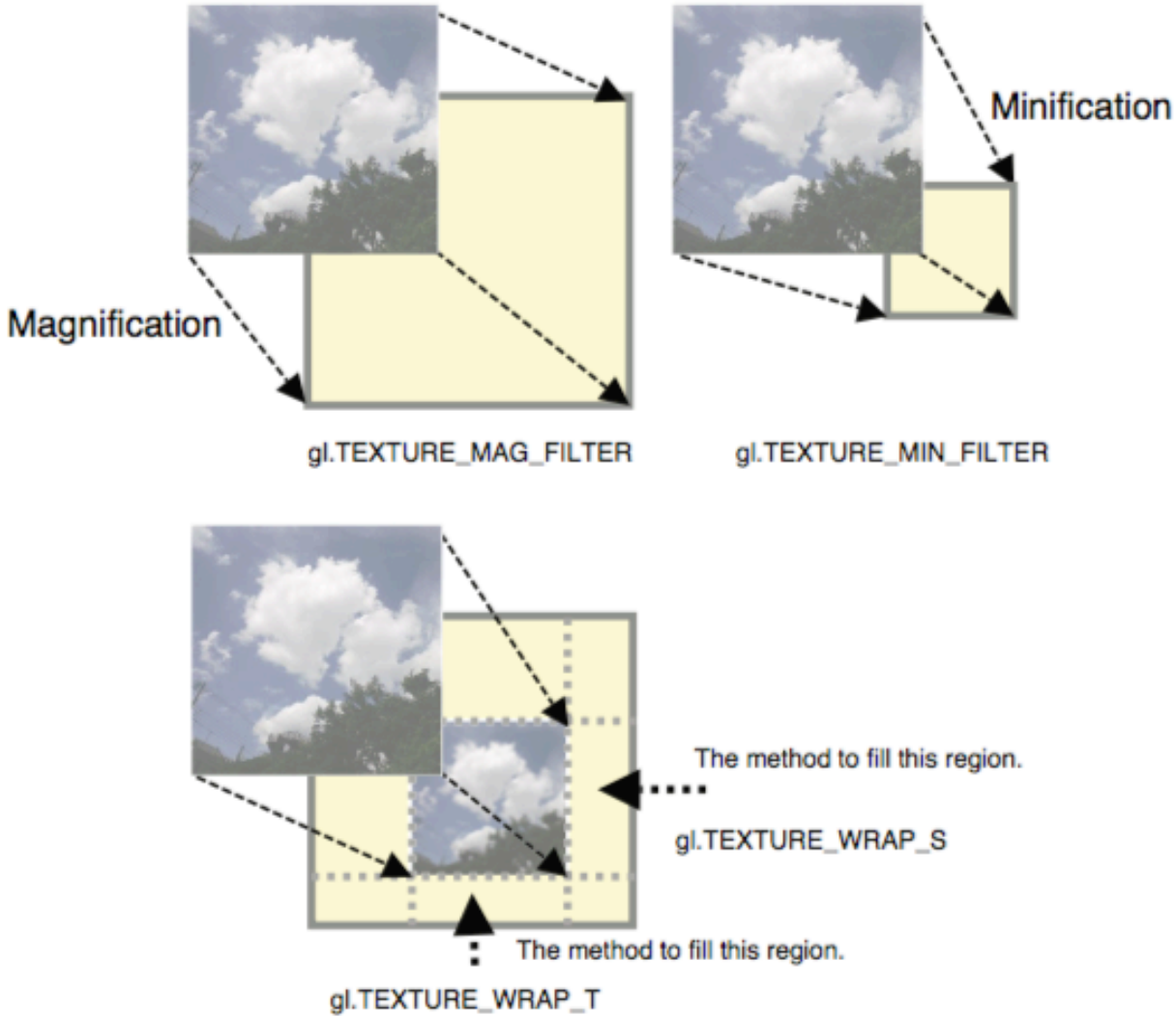
Erorrs:

- INVALID_ENUM: *target* is none of the preceding values.
- INVALID_OPERATION: no texture object is bound to *target.*

Nombre del parámetro de textura

- **Magnification method** (gl.TEXTURE_MAG_FILTER): mapear una textura en una figura con un área mayor. Por ejemplo, para mapear una textura de 16x16 sobre una figura de 32x32, WebGL necesita rellenar el hueco entre los texels y este parámetro especifica el método a utilizar para rellenar dicho hueco.
- **Minification method** (gl.TEXTURE_MIN_FILTER): mapear una textura en una figura con un área menor. Por ejemplo, para mapear una textura de 32x32 sobre una figura de 16x16, WebGL necesita desechar texels y este parámetro especifica el método a utilizar para determinar los texels a eliminar.
- **Wrapping method on the left and right side** (gl.TEXTURE_WRAP_S): como rellenar la regiones a la izquierda y a la derecha cuando se mapea una textura en una subregión de la figura.
- **Wrapping method on top and bottom** (gl.TEXTURE_WRAP_T): igual que el anterior pero para rellenar las restantes zonas por arriba y por abajo.

Nombre del parámetro de textura



Parámetros de textura y valores por defecto

gl.TEXTURE_MAG_FILTER ➔ Texture magnification ➔ gl.LINEAR
gl.TEXTURE_MIN_FILTER ➔ Texture minification ➔ gl.NEAREST_MIPMAP_LINEAR
gl.TEXTURE_WRAP_S ➔ Texture wrapping in s-axis ➔ gl.REPEAT
gl.TEXTURE_WRAP_T ➔ Texture wrapping in t-axis ➔ gl.REPEAT

- **Parámentros específicos de MAG y MIN:**
  - gl.NEAREST: utiliza el valor del texel más cercano al centro del pixel que se va a texturizar.
  - gl.LINEAR: utiliza la media ponderada de los cuatro téxeles que están más cerca del centro del píxel que se va a texturizar.

- **Parámentros específicos de WRAP_S y WRAP_T:**
  - gl.REPEAT: usa una imagen de textura repetidamente.
  - gl.MIRRORED_REPEAT: textura reflejo-repetidamente.
  - gl.CLAMP_TO_EDGE: usa el color del borde de una imagen de textura

Asignar imagen de textura a objeto de textura

```
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
```

gl.texImage2D(target, level, internalformat, format, type, image): set the image specified by *image* to the texture object bound to *target.*

Parameters

- target: specifies gl.TEXTURE_2D or gl.TEXTURE_CUBE_MAP
- level: specifies as 0.
- internalformat: specifies the format of the texel data.
- format: this must be specified using the same value as *internalformat*.
- type: specifies the data type of the texel data
- image: Specifies an Image object containing an image to be used as a texture.

Return value: none.

Errors:

- INVALID_ENUM: target is none of the above values.
- INVALID_OPERATION: no texture object is bound to target.

Formato y tipo de dato del texel

```
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
```

Formato:

gl.RGB ➔ Red, green, blue.
gl.RGBA ➔ Red, green, blue, alpha.
gl.ALPHA ➔ 0.0, 0.0, 0.0, alpha.
gl.LUMINANCE ➔ L, L, L, 1 L: luminance.
gl.LUMINANCE_ALPHA ➔ L, L. L. alpha.

Tipo:

gl.UNSIGNED_BYTE ➔ Unsigned byte format. Each color component has 1 byte.
gl.UNSIGNED_SHORT_5_6_5 ➔ RGB: has 5, 6, and 5 bits, respectively.
gl.UNSIGNED_SHORT_4_4_4_4 ➔ RGBA: has 4, 4, 4, and 4 bits, respectively.
gl.UNSIGNED_SHORT_5_5_5_1 ➔ RGBA: RGB has 5 bits, and A has 1 bit.

Recuperar el color de Texel en el Fragment Shader

```
gl_FragColor = texture2D(uSampler, vTexCoord);
```

vec4 texture2D(sampler2D sampler, vec2 coord): retrieve a texel color at the texture coordinates specified by *coord* from the texture image specified by *sampler.*

Parameters

- sampler: specifies the texture unit number.
- coord: specifies the texture coordinates.

Return value: the texel color (vec4) for the coordinates. The color format changes according to the *internalformat* specified by gl.texImage2D(). If the texture image is not available for some reason, this function returns (0.0, 0.0, 0.0, 1.0).