

# Texturas

[mleguen@ucsp.edu.pe](mailto:mleguen@ucsp.edu.pe)

# Problemática

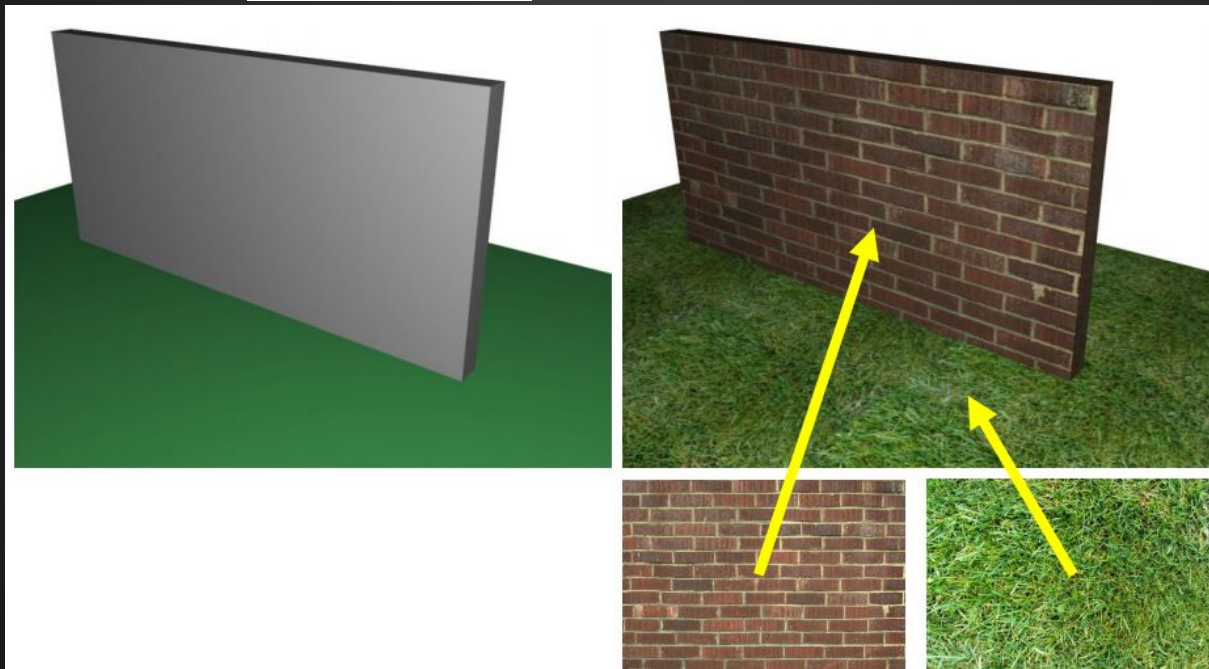
Modelo Phong : tomamos en cuenta la luz y el material



Falta de realismo en una escena que solo utiliza el modelo de iluminación de Phong

# Solución

Utilizamos texturas

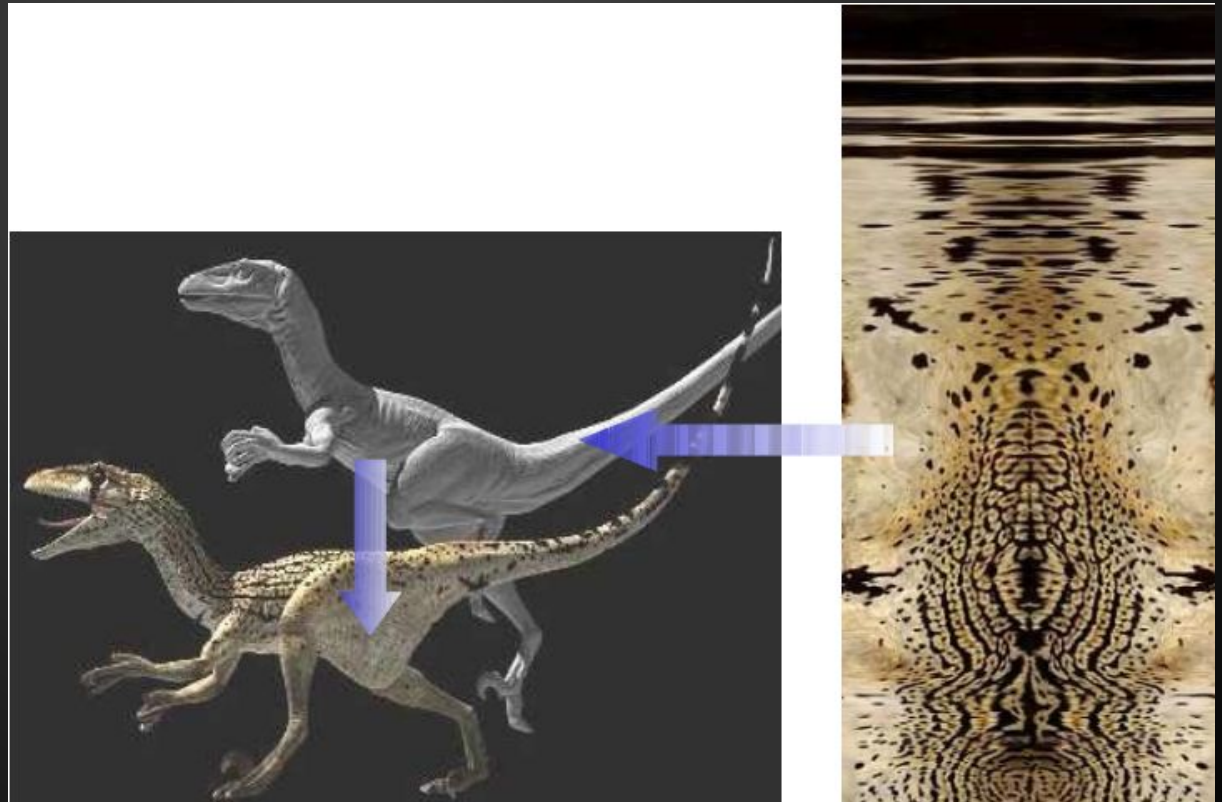


Mapeado de texturas (“Texture mapping”)

Mapeado de texturas sobre primitivas  
geométricas

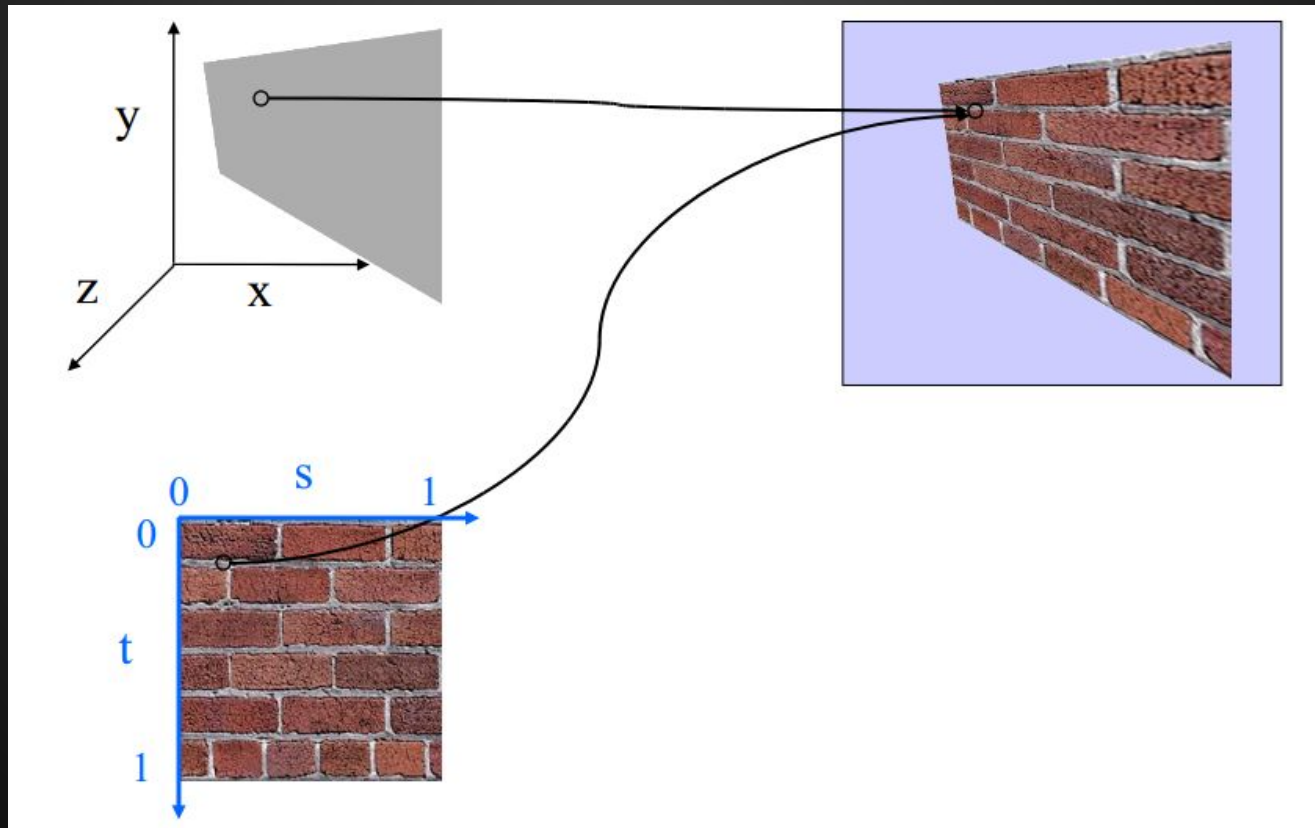
# Objetivo

- Simular materiales (madera, piedra, piel ... )
- Reducir la complejidad (cantidad de polígonos ) de los objetos 3D
- Simular superficies reflejantes



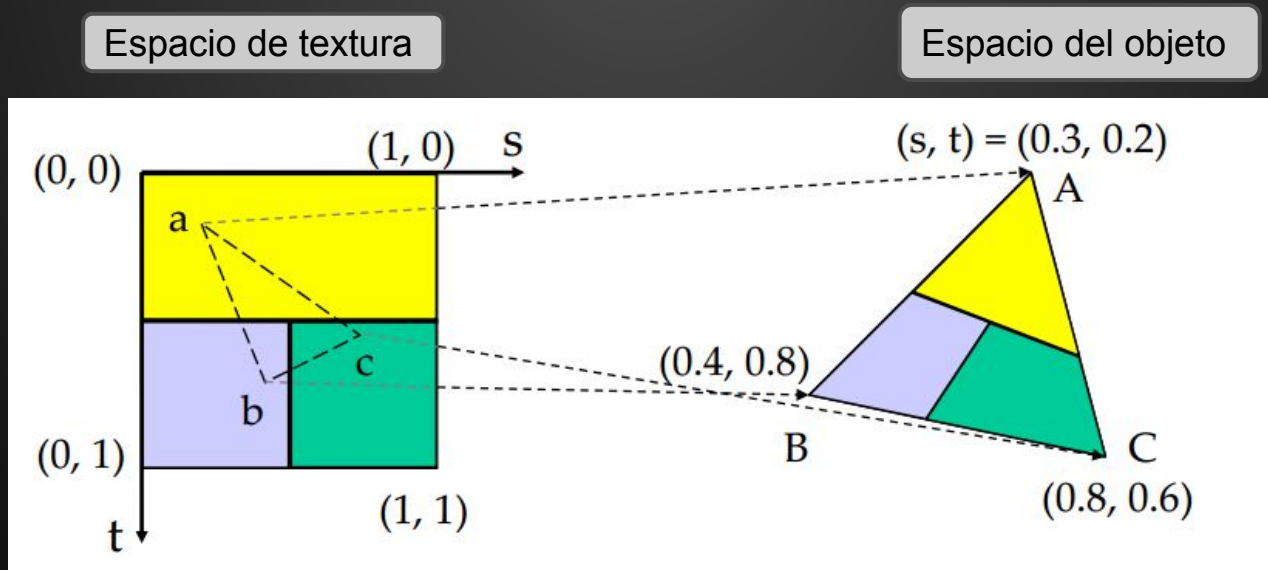
# Principio

- Mapeo de texturas
  - Coordenadas de texturas



# Principio

- Las texturas son imágenes que pueden ser
  - 1D : coordenadas de texturas (s)
  - 2D : coordenadas de texturas (s,t)**
  - 3D : coordenadas de texturas (s,t,r)
  - 4D : coordenadas de texturas (s,t,r,q)
- Para una textura 2D, un punto en la imagen es obtenido por sus coordenadas (s,t), la esquina superior derecha (0,0) y la esquina inferior izquierda (1,1)



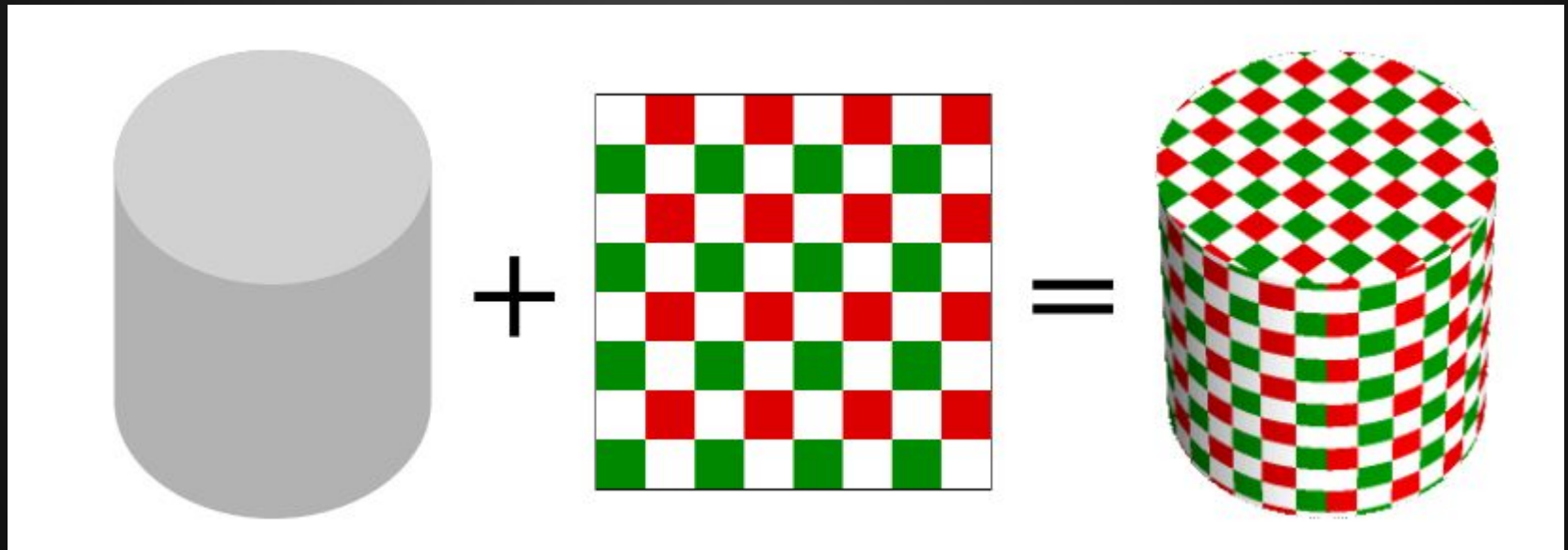
# Principio

- Observaciones :
  - Un píxel de una textura se llama texel
  - En versiones antiguas de OpenGL el tamaño de texturas tenían que ser en potencia de 2
  - Las dimensiones de las texturas son limitadas por las tarjetas gráficas.



# Parametrización

- Como definir las coordenadas de texturas de los vértices del objeto 3D que tenemos que texturar ?



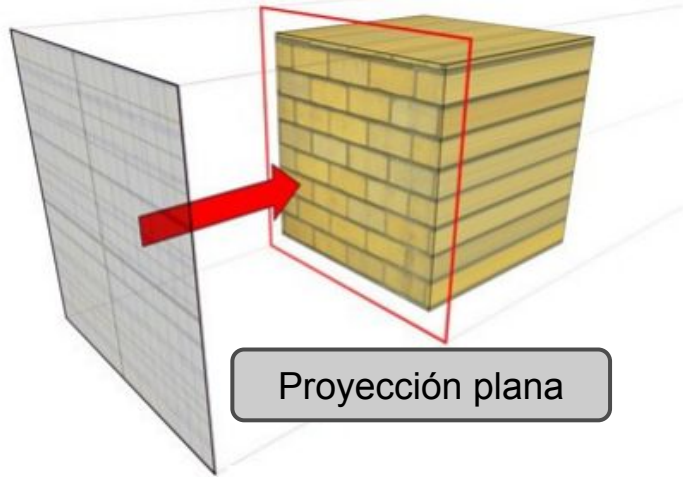
Objeto 3D

Textura

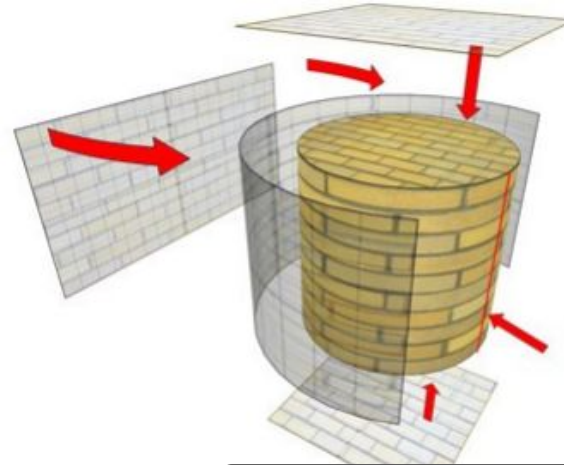
Objeto texturado



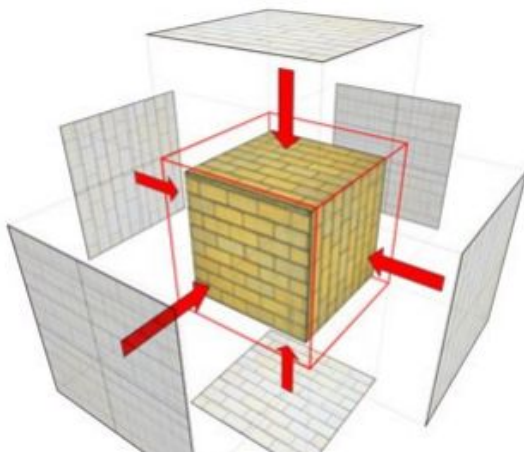
# Proyección



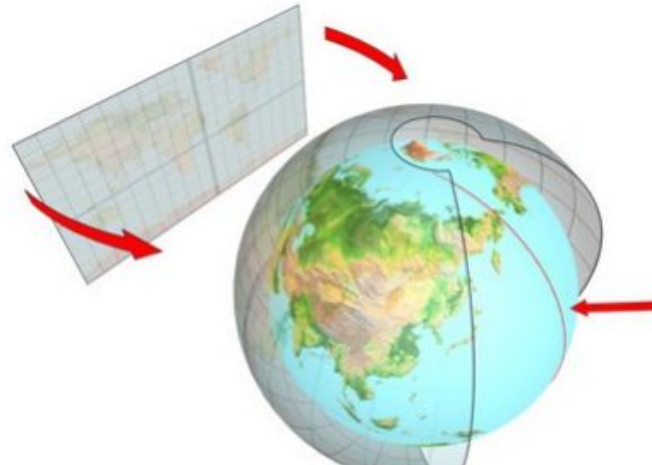
Proyección plana



proyección cilíndrica

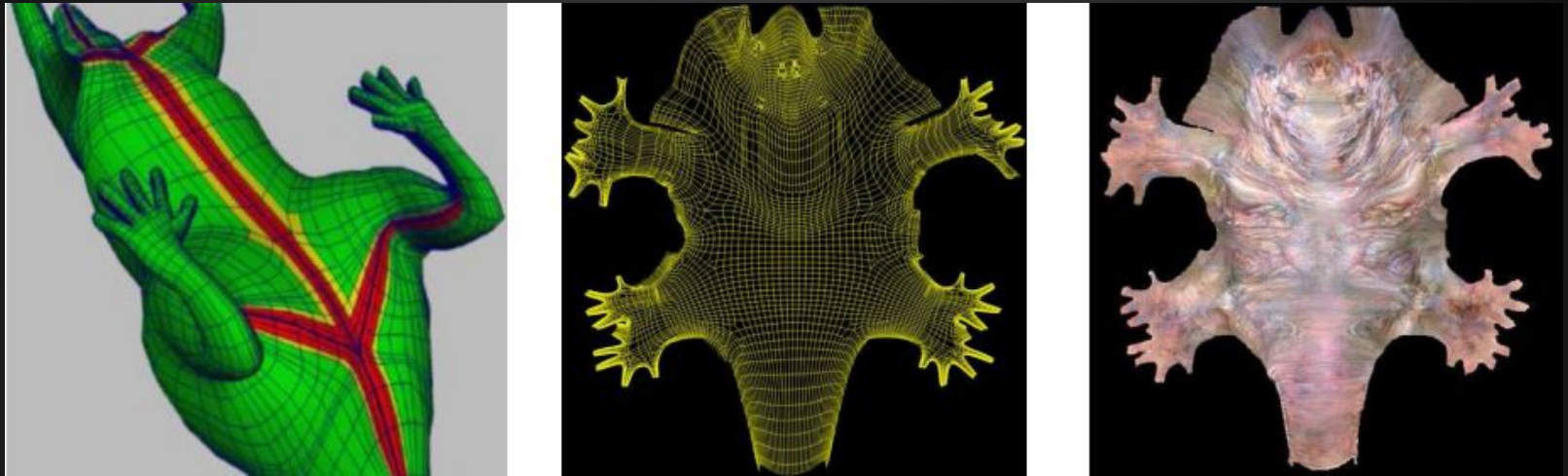


Proyección cubica



Proyección esferica

# Mapeado de texturas

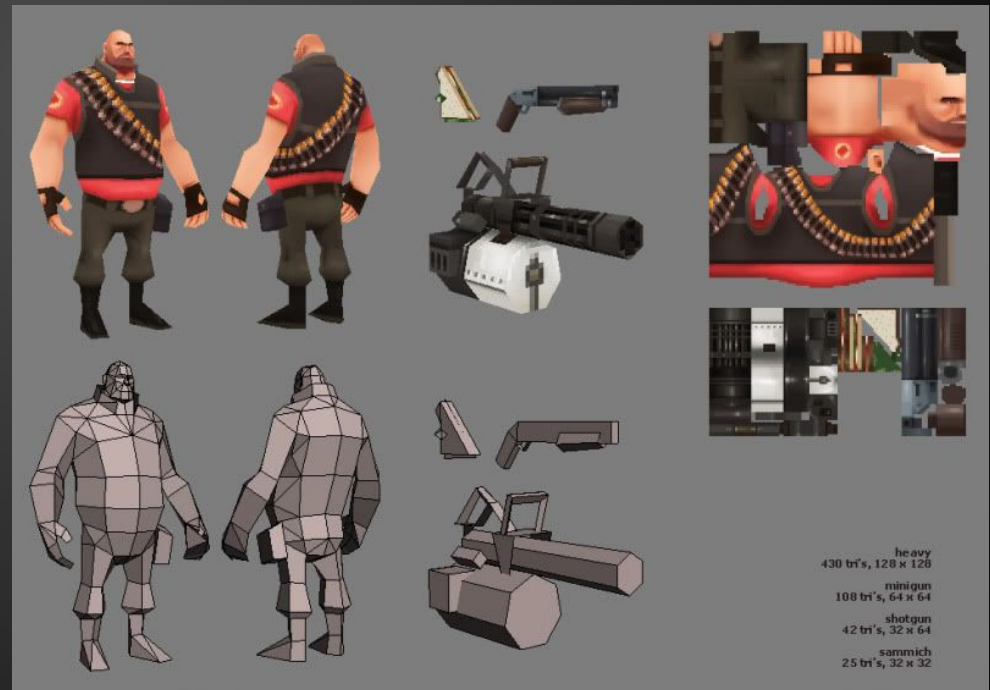


Se despliega la malla del modelo 3D, dibujado en la imagen 2D de la textura con un algoritmo que trata de conservar la superficie de los triángulos. Después, un artista dibuja la textura encima usando la malla como una guía.



# Atlas de texturas

En el espacio de la textura, las diferentes partes de la textura son disjuntos. Corresponden a grupos de triángulos del modelo 3D. Hay que repartir los grupos de manera a optimizar la ocupación del espacio de la textura.



# 3D painting

Algunos software de creación de texturas permiten dibujar directamente sobre el modelo 3D. Estos software crean o despliegan automáticamente la textura.

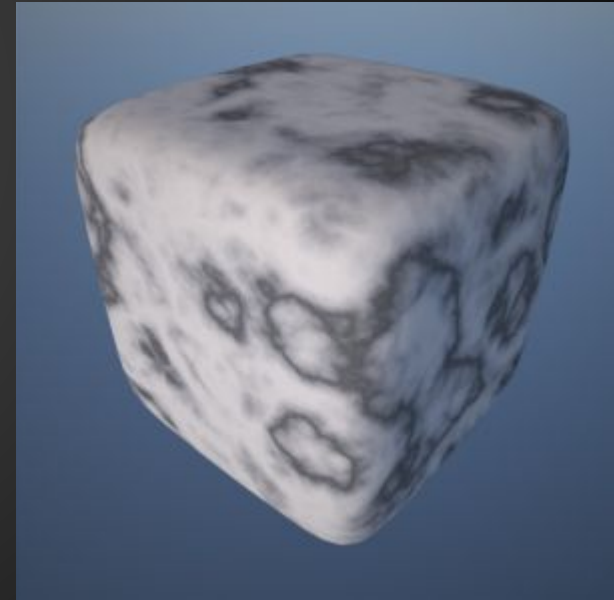




# Procedural texture

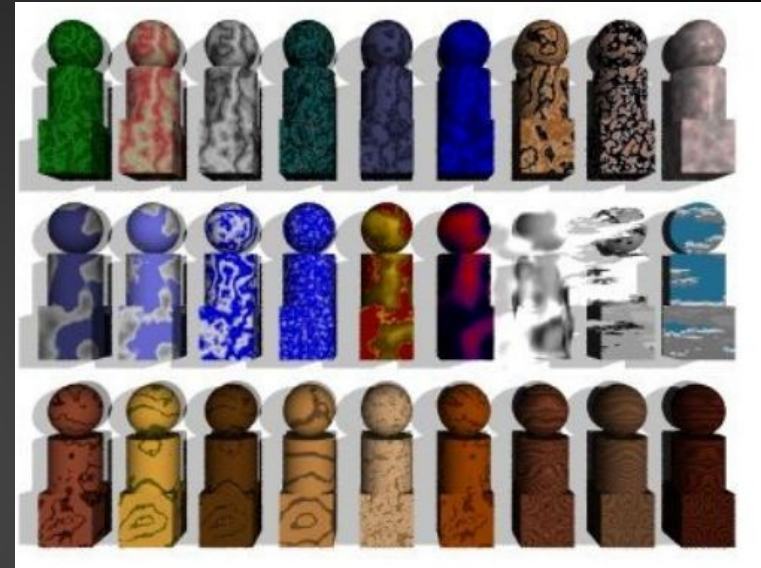
Las texturas pueden ser de dos tipos :

- Raster : imagen compuesta de píxeles (más común)
- Procedural = función matemática
  - Calculamos el color del píxel en todo punto de la superficie gracias a una función  $f(x,y)$  o en todo de un volumen gracias a una función  $f(x,y,z)$ .
- Podemos simplemente escribir estas funciones gracias a los shaders



# Procedural texture

- Texturas procedurales de madera, piedra, nubes..
- Las texturas procedurales son utilizadas en software de diseño 3D (3Ds max, Blender ...)



# Procedural texture

- Ventajas de texturas procedurales
  - Se necesita muy poca memoria para almacenar la textura : código de las funciones
  - Evita problemas de despliegue de texturas
- Desventajas de texturas procedurales
  - El cálculo del color en un punto puede ser largo dependiendo de la complejidad de la función utilizada
  - Determinar los parámetros de estas funciones no es necesariamente simple e intuitivo



# Utilizar texturas en OpenGL

3 etapas :

1. Especificar una textura
  - a. leer o generar una imagen
  - b. convertirla en una textura
  - c. activar el mapeo de texturas
2. Asignar las coordenadas de texturas a cada vértice de objeto 3D
3. Especificar los parámetros de texturas
  - a. Wrapping, filtering...

# Especificar una textura

- Leer o generar una imagen

```
BYTE *img;
```

```
int dim_x, dim_y;
```

```
GLuint texture;
```

```
glGenTextures(1, &texture);
```

```
img = load_pic( "image.tga", &dim_x, &dim_y );
```

# Especificar una textura

- Explicación

- En OpenGL cada textura es referenciada por un índice (entero). Para obtener un índice tenemos que utilizar esta función :
  - `glGenTextures(GLuint n, GLuint *text_array);`
  - Crean índices de texturas y lo coloca en la tabla de enteros : `text_array`.

# Especificar una textura

- Explicación

- Si queremos obtener 10 índices de texturas :

```
GLuint tab_text[10];
```

```
glGenTextures(10, tab_text);
```

- Si queremos una sola textura

```
GLuint texture;
```

```
glGenTextures(1, &texture);
```

# Especificar una textura

- Crear una textura
  - Las texturas deben ser almacenadas en la memoria del GPU.
    - Cuando cargamos una imagen para convertirla en textura necesitamos transferirla a la memoria del GPU.

```
glBindTexture(GL_TEXTURE_2D, texture);
```

```
glTexImage2D( GL_TEXTURE_2D, 0, 3, dim_x, dim_y,  
              0, GL_RGB, GL_UNSIGNED_BYTE, img);
```

- La función `glBindTexture` permite fijar el índice de la textura : Esta textura será utilizada para todas las proximas funciones que tienen que ver con texturas (mapeo, parámetros, etc...) hasta la proxima llamada de `glBindTexture`

# Especificar una textura

`glTexImage2D( target, level, components, w, h,  
border, format, type, *texels );`

- target : GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D
- level : 0 sin mip-mapping
- components : cantidad de elemento por pixel : 3 -> RGB
- w, h : dimensiones de la textura
- border : borde suplementario al imagen
- format : GL\_RGB, GL\_RGBA, ...
- type : tipo de datos de los texels
- texels : tabla de texels (datos)

# Especificar una textura

`glTexImage2D( target, level, components, w, h,  
border, format, type, *texels );`

- target : GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D
- level : 0 sin mip-mapping
- components : cantidad de elemento por pixel : 3 -> RGB
- w, h : dimensiones de la textura
- border : borde suplementario al imagen
- format : GL\_RGB, GL\_RGBA, ...
- type : tipo de datos de los texels
- texels : tabla de texels (datos)



# Activar el mapeo de texturas

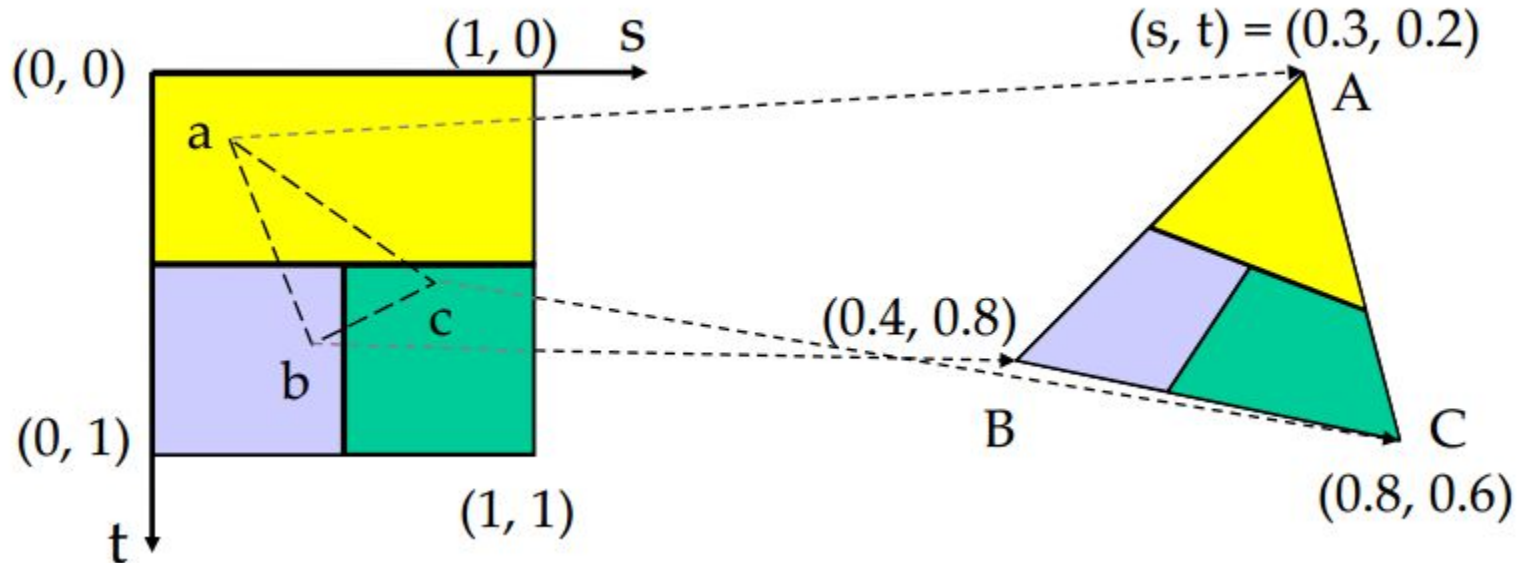
- Activar el mapeo de texturas
  - `glEnable(GL_TEXTURE_2D)`
- Activar el mapeo de texturas
  - `glEnable(GL_TEXTURE_2D)`
- Podemos como para el `glEnable(GL_LIGHTING)` llamar estas funciones en cada momento para activar/desactivar el uso de texturas (para por ejemplo visualizar un ejemplo sin texturas).

# Asignar coordenadas de texturas a un los vértices de un objeto 3D

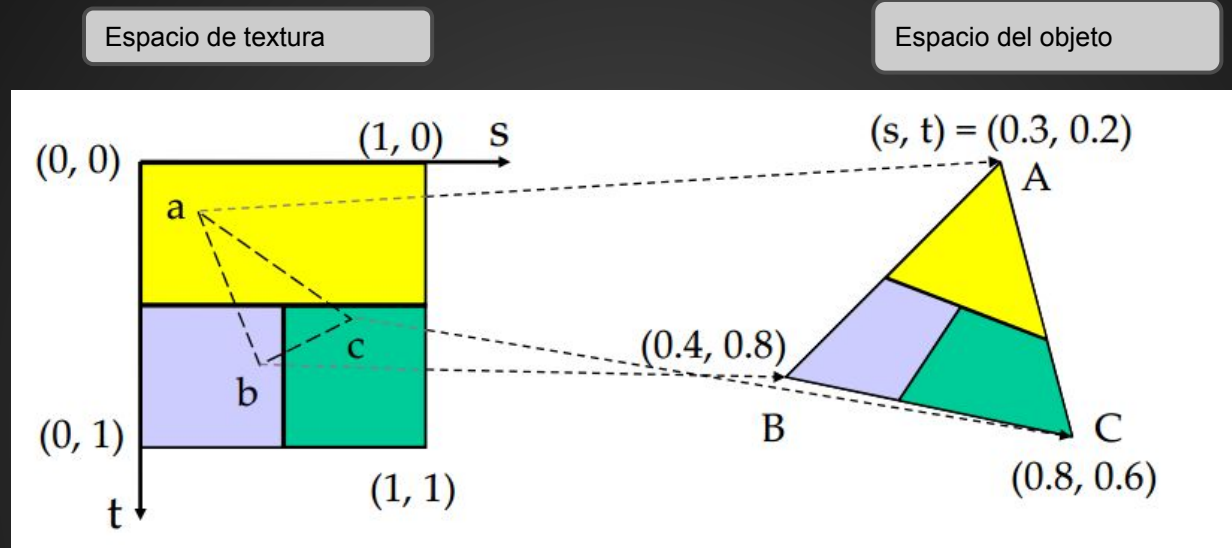
Para mapear una textura sobre un objeto geométrico, se necesita atribuir a cada vértice del objeto coordenadas 2D de texturas normalizadas.

Espacio de textura

Espacio del objeto



# Asignar coordenadas de texturas a un los vértices de un objeto 3D



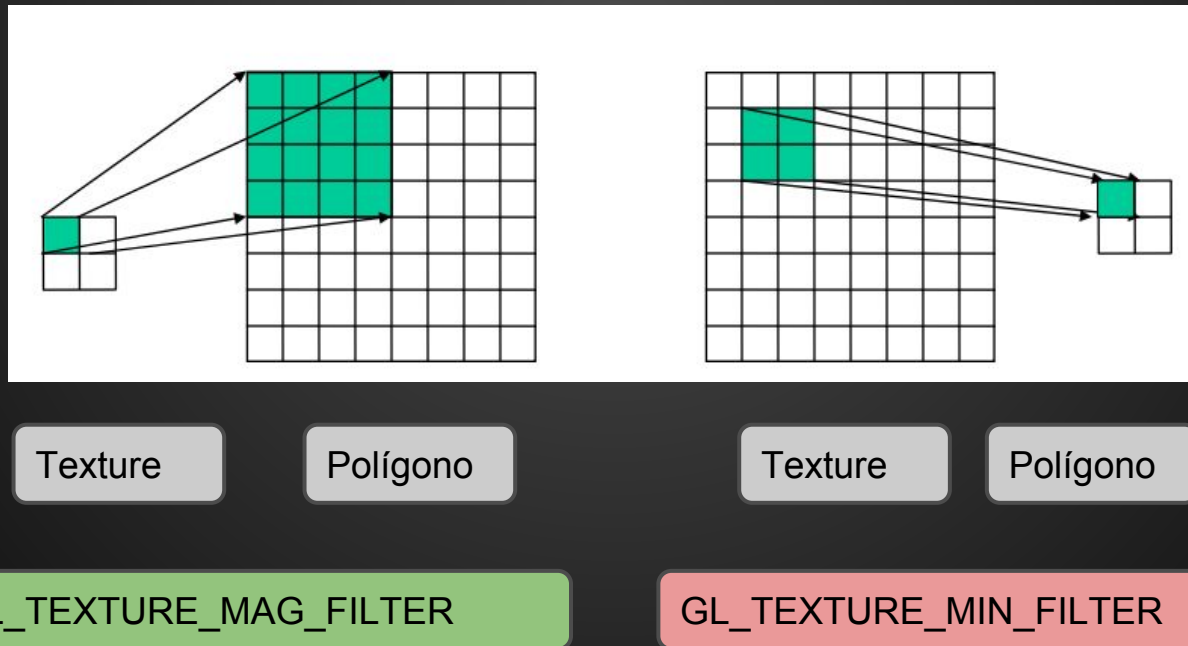
```
glBindTexture(GL_TEXTURE_2D, texture);
glBegin(GL_TRIANGLES);
    glTexCoord2f(0.3f, 0.2f);
    glVertex3f(10.0f, 12.0f, 0.0f);
    glTexCoord2f(0.4f, 0.8f);
    glVertex3f(4.0f, 6.0f, 0.0f);
    glTexCoord2f(0.8f, 0.6f);
    glVertex3f(15.0f, 2.0f, 0.0f);
glEnd();
```

# Parámetros de textura

- Modo de filtración
  - Reducción, ampliación
- Modo de repetición
  - repetir, truncar
- Función de texturas
  - cómo mezclar el color del objeto con su textura

# Filtros

- Reducción, ampliación
- Las texturas y los objetos texturizados casi nunca tienen el mismo tamaño. OpenGL define filtros para definir como un texel debe ser reducido o agrandado para corresponder al tamaño de un pixel



# Filtros

- `glTexParameterf(target, type, mode);`

`target` : `GL_TEXTURE_2D`, ...

`type` : `GL_TEXTURE_MIN_FILTER`, `GL_TEXTURE_MAG_FILTER`

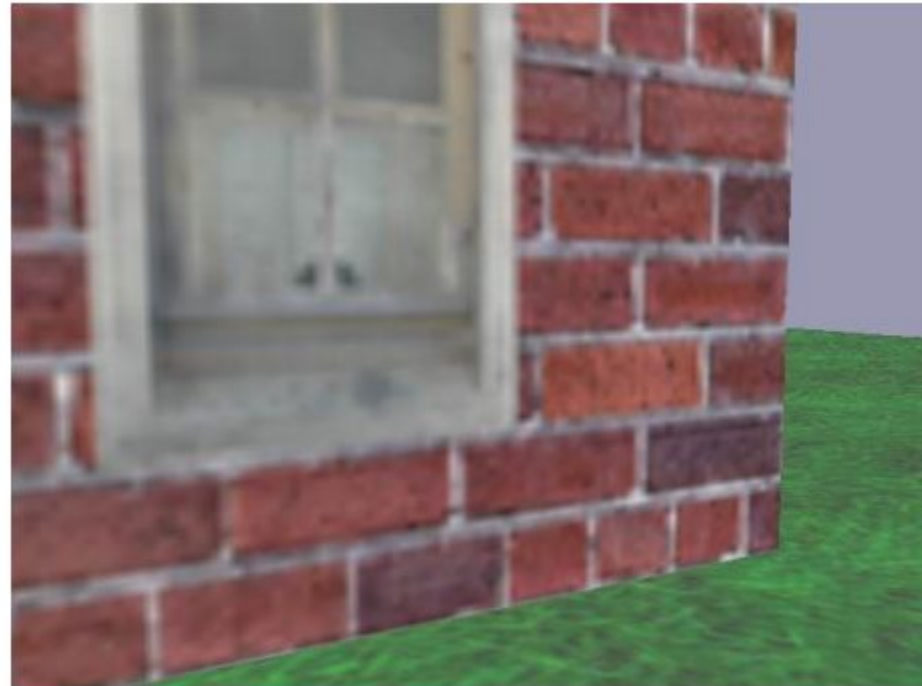
`mode` : `GL_NEAREST`, `GL_LINEAR`

- `GL_NEAREST` : El color del pixel es obtenido a partir del texel el más cerca
- `GL_LINEAR` : El color del pixel es calcular por interpolación entre los texels más cerca partir de los texels vecinos

# Filtros



**GL\_NEAREST**



**GL\_LINEAR**



# Filtros

Ejemplo :

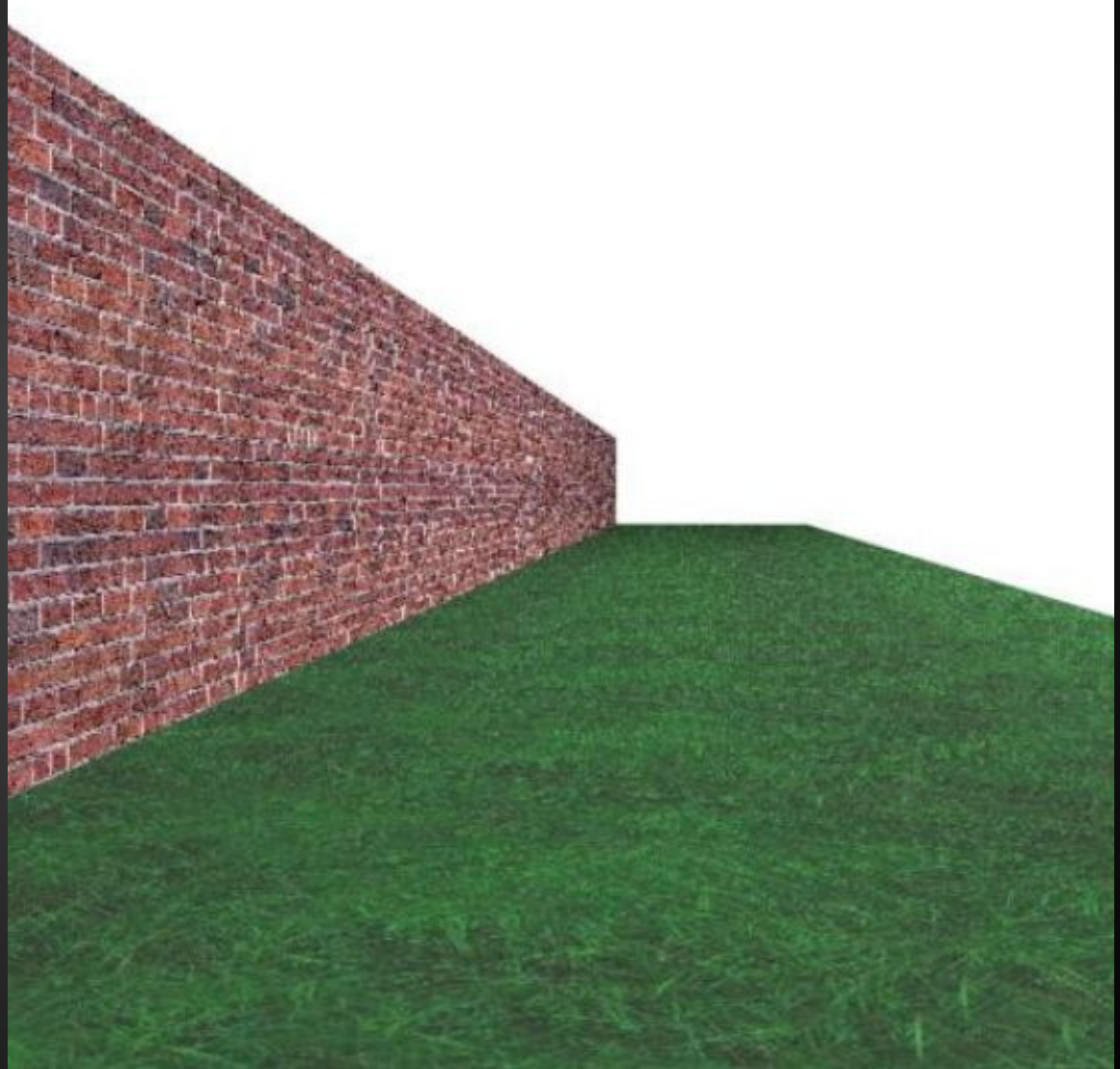
```
glBindTexture(GL_TEXTURE_2D, texture);
```

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

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

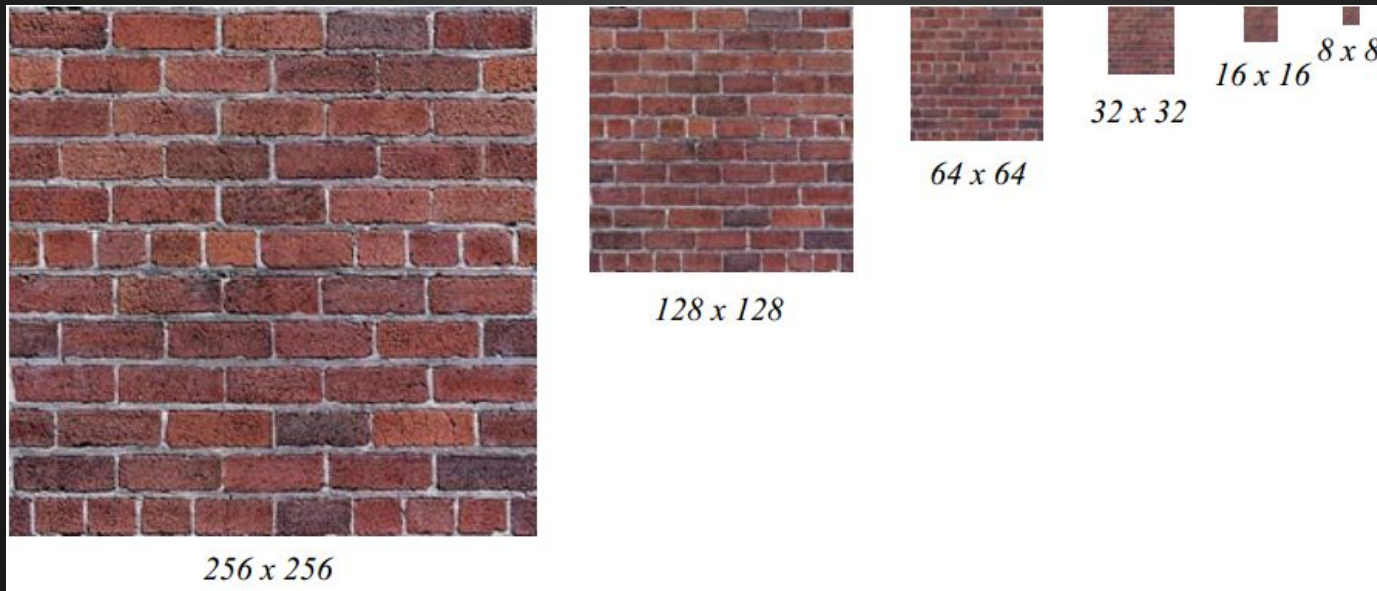
# Filtros

Mip-mapping : problema de aliasing



# Filtros

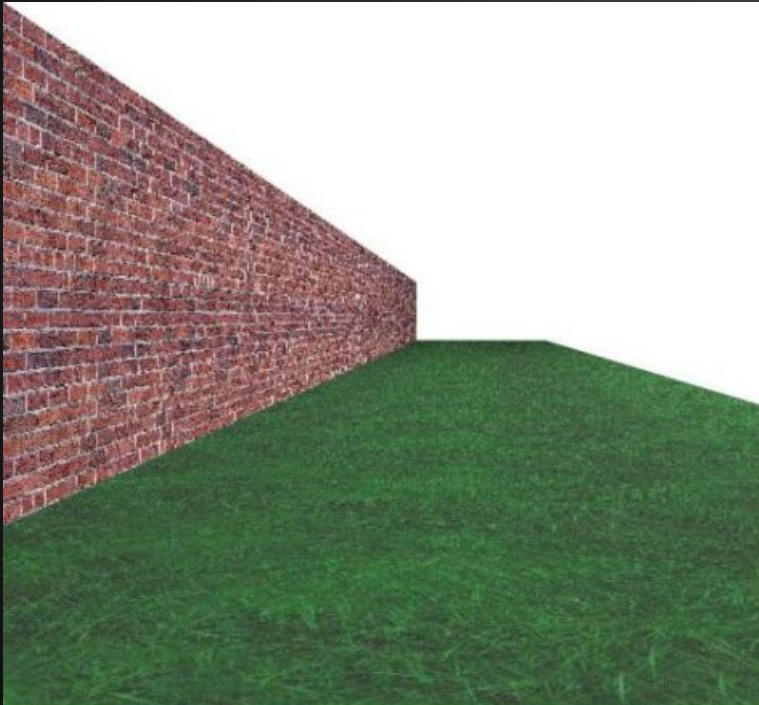
Solución : La técnica de Mip-mapping consiste en pre-calcular varias versiones reducidas de la textura.



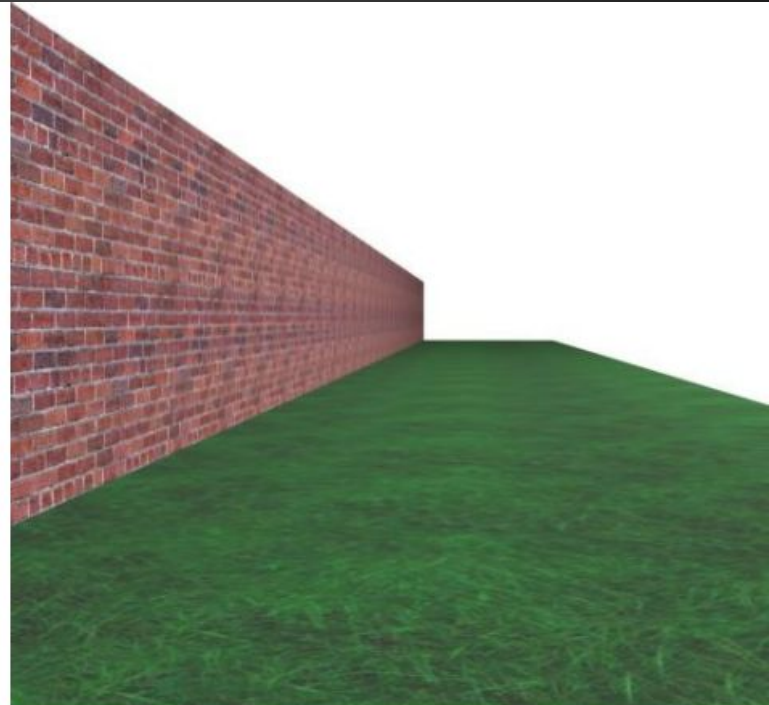
Adapta el nivel de detalle en función a la distancia

# Filtros

Mip-mapping :



sin mip-mapping



con mip-mapping

# Filtros

Mip-mapping : Una función de OpenGL permite de construir automáticamente las diferentes texturas de mip-map.

```
gluBuild2DMipmaps(GL_TEXTURE_2D, 3, width, height, GL_RGB, GL_UNSIGNED_BYTE, img);
```

Hay entonces que indicar el modo de filtración de la textura indicando que la utilizaremos como mip-map

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

# Modo de repetición

Este modo indica lo que debemos hacer cuando una coordenada de textura sale de rango  $(0,1)$

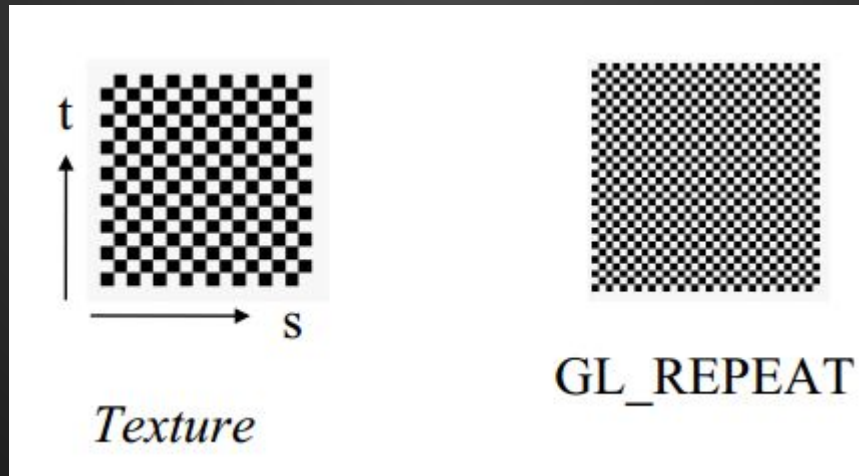
- Dos posibilidades
  - repetir (repeat)
  - truncar (clamp)



# Modo de repetición

- Repetir
  - Si el modo GL\_REPEAT es utilizado para las coordenadas  $>1$  o  $<0$

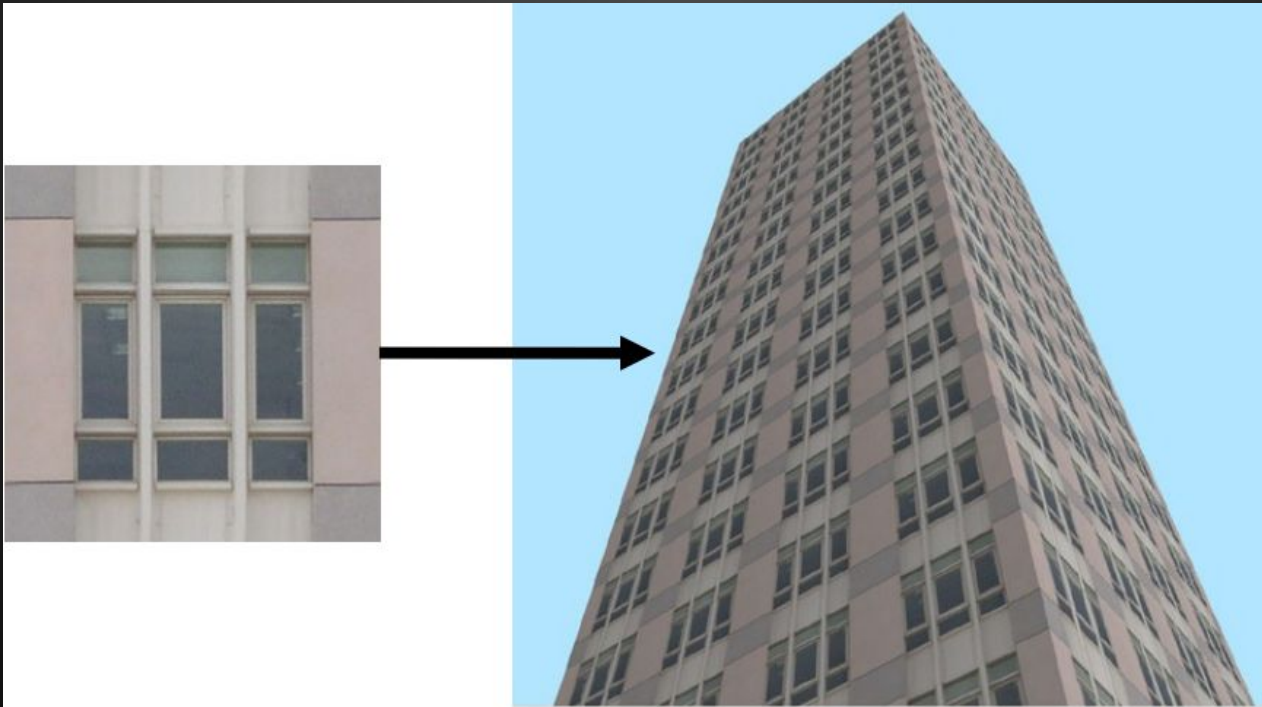
```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
```





# Modo de repetición

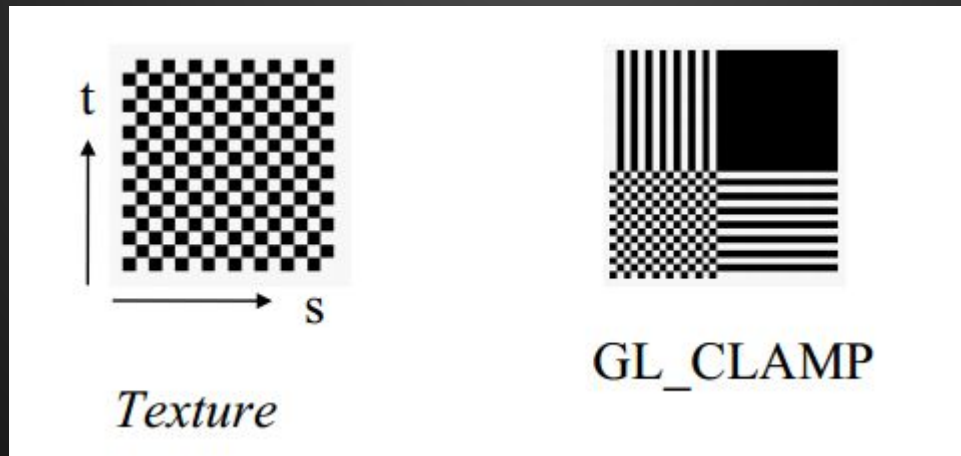
- Ejemplo : Repetir
  - Tenemos que utilizar una textura que se repita naturalmente



# Modo de repetición

- Truncar
  - Si el modo GL\_CLAMP es utilizado para las coordenadas  $>1$  o  $<0 \rightarrow 1$  o  $0$  es utilizado

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



# Funciones de texturas

- Controla de qué manera mezclamos la textura el color del objeto

```
glTexEnvf( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, param );
```

- param :
  - GL\_DECAL : reemplaza el color por el texel
  - GL\_MODULATE : multiplica el color del texel por el color

# Funciones de texturas



- GL\_DECAL :  
reemplaza el color  
por el texel (bye  
bye Phong)

- GL\_MODULATE :  
multiplica el color  
del texel por el  
color

# Vertex shader

```
#version 330 core
```

```
layout(location = 0) in vec3 vertexPosition;  
layout(location = 1) in vec2 vertexUV;
```

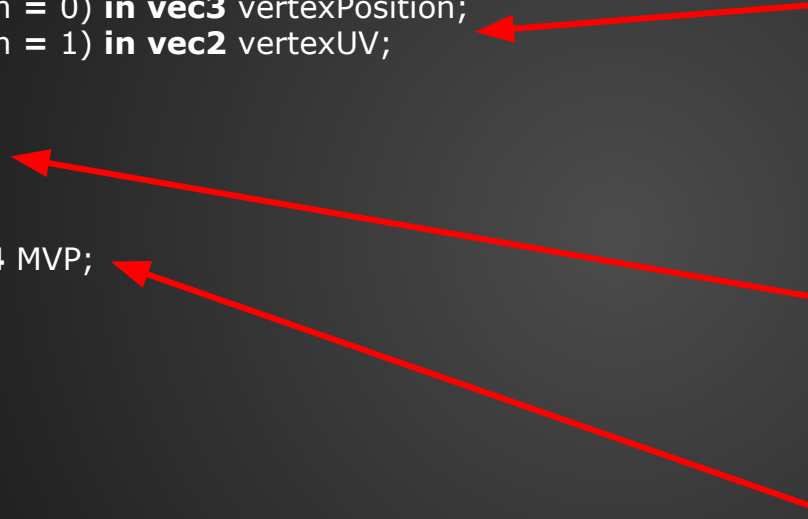
```
out vec2 UV;
```

```
uniform mat4 MVP;
```

```
void main(){
```

```
    gl_Position = MVP * vec4(vertexPosition,1);  
    UV = vertexUV;  
}
```

Datos de un vértice (posición  
+ coordenadas de textura)



coordenadas de textura que  
queremos mandar al  
fragment shader

Matriz MVP

# Fragment shader

```
#version 330 core
```

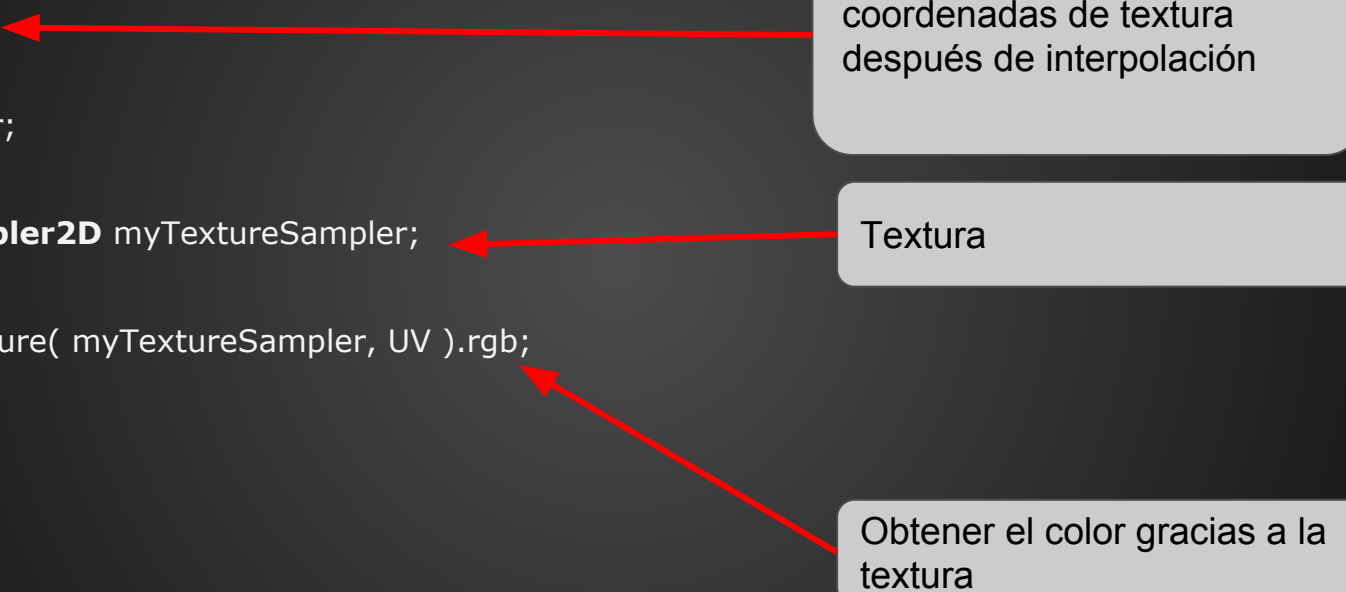
```
in vec2 UV;
```

```
out vec3 color;
```

```
uniform sampler2D myTextureSampler;
```

```
void main(){  
    color = texture( myTextureSampler, UV ).rgb;  
}
```

coordenadas de textura  
después de interpolación



Textura

Obtener el color gracias a la  
textura

# Textura + iluminación

- Utilizar el color de la textura como color del material difuso

```
vec3 MaterialDiffuseColor = texture( myTextureSampler, UV ).rgb;
```

# Mandar la textura como uniforme

- Cargar la textura

```
GLuint Texture = loadtexture("texture.png");
```

- Recuperar el id del uniform en el shader

```
GLuint TextureID = glGetUniformLocation(programID, "myTextureSampler");
```

- En el bucle de rendering
  - Vincular la textura la unidad 0 de texturas

```
glActiveTexture(GL_TEXTURE0);
```

```
glBindTexture(GL_TEXTURE_2D, Texture);
```

- Mandar la textura en “myTextureSampler”

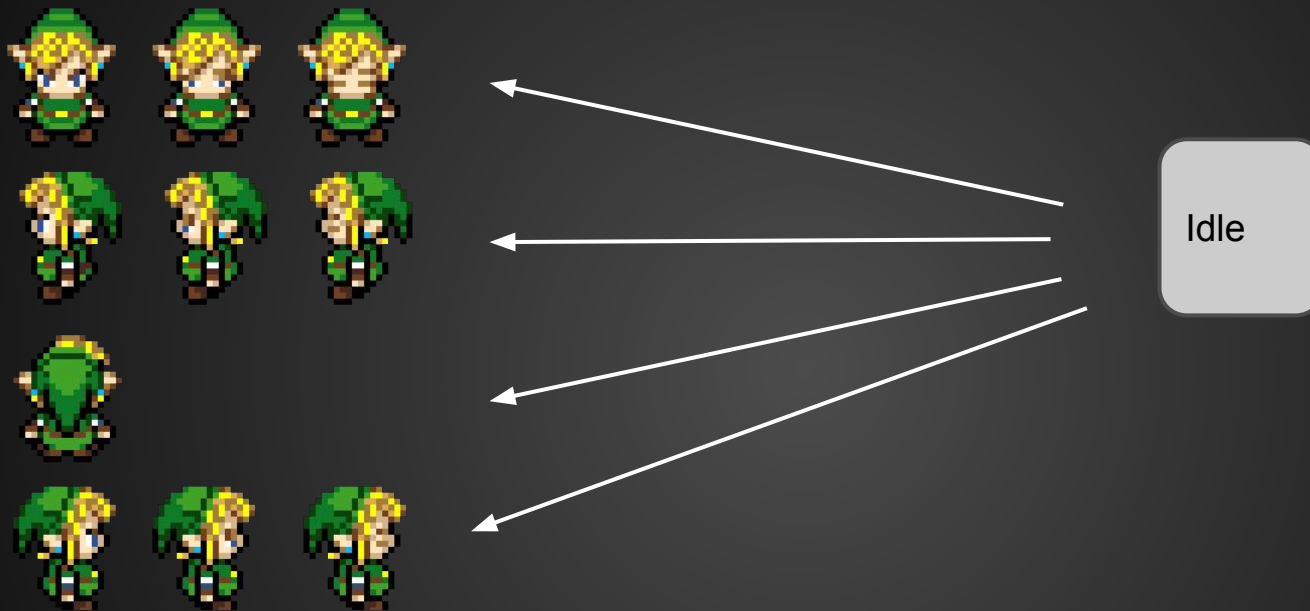
```
glUniform1i(TextureID, 0);
```



# Sprite



# Sprite - Animación



# Sprite - Animación



Abajo

Izquierda

Arriba

Derecha

# Sprite - Animación

- Recortar un personaje de la textura para para mapearlo sobre un simple rectángulo
- Cambiar las coordenadas de textura según el movimiento deseado



# Sprite - Animación

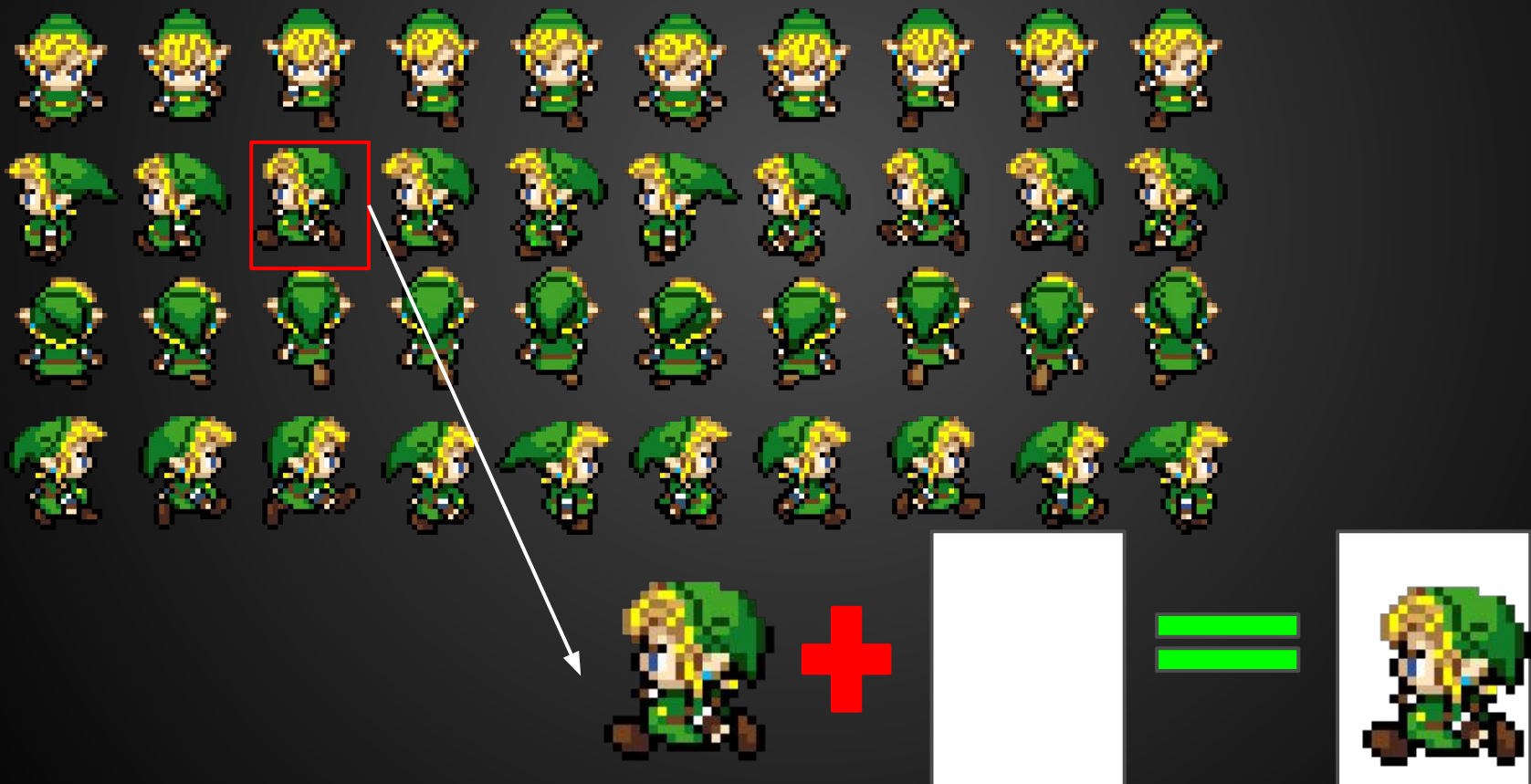




# Sprite - Animación

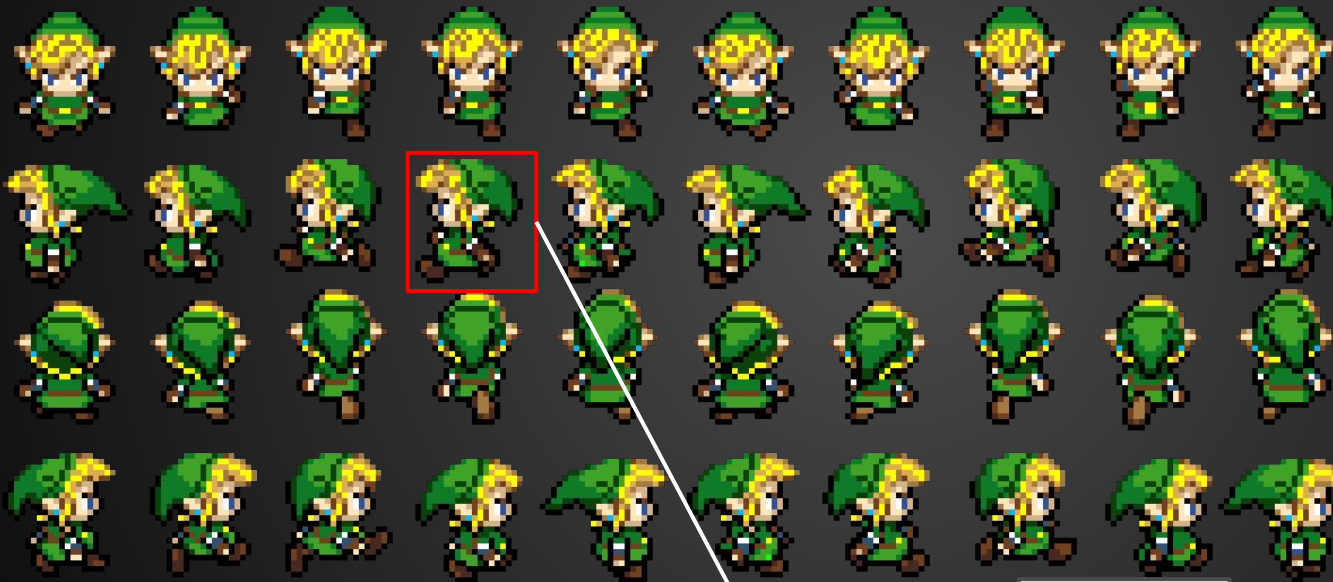


# Sprite - Animación





# Sprite - Animación



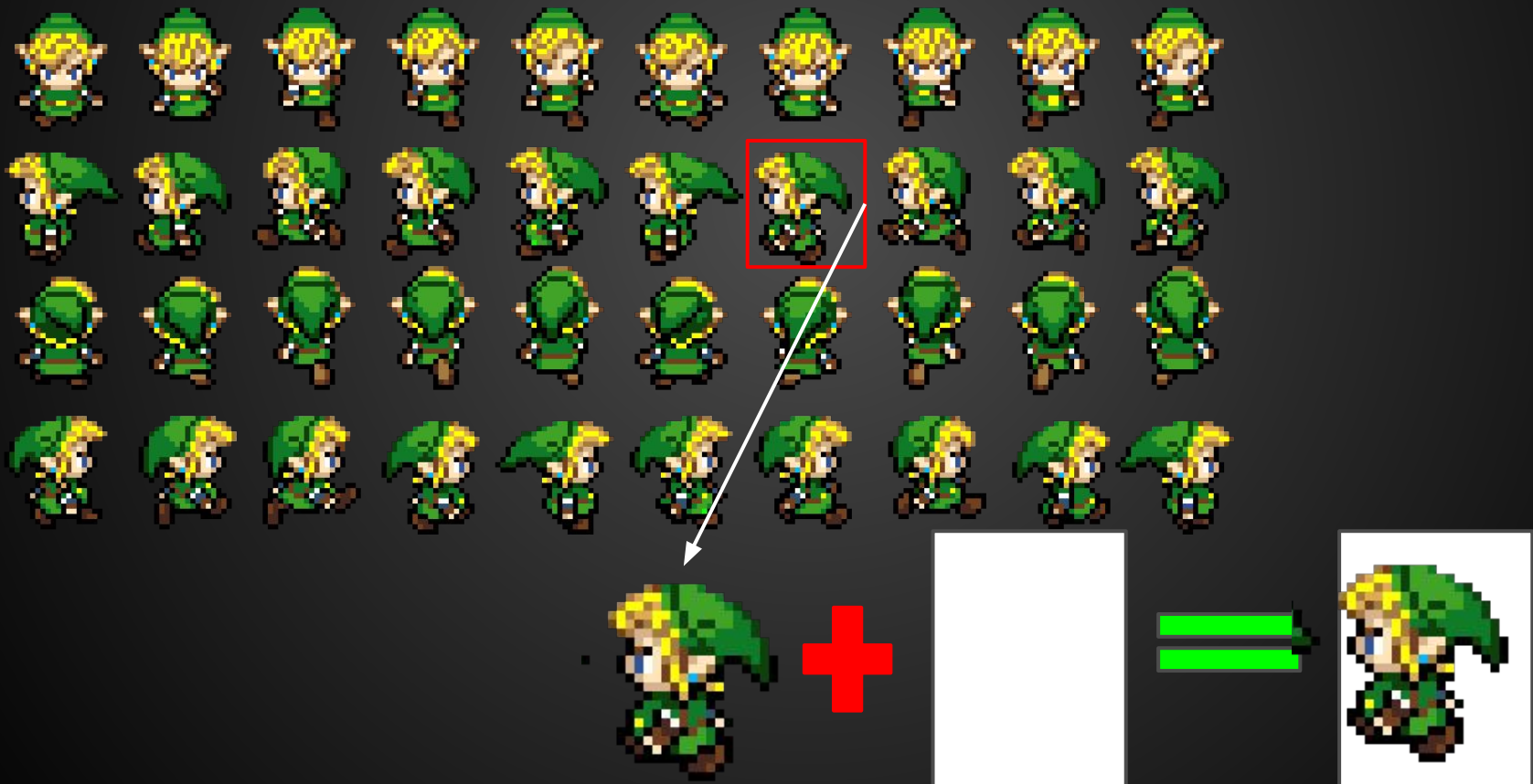
# Sprite - Animación



# Sprite - Animación



# Sprite - Animación





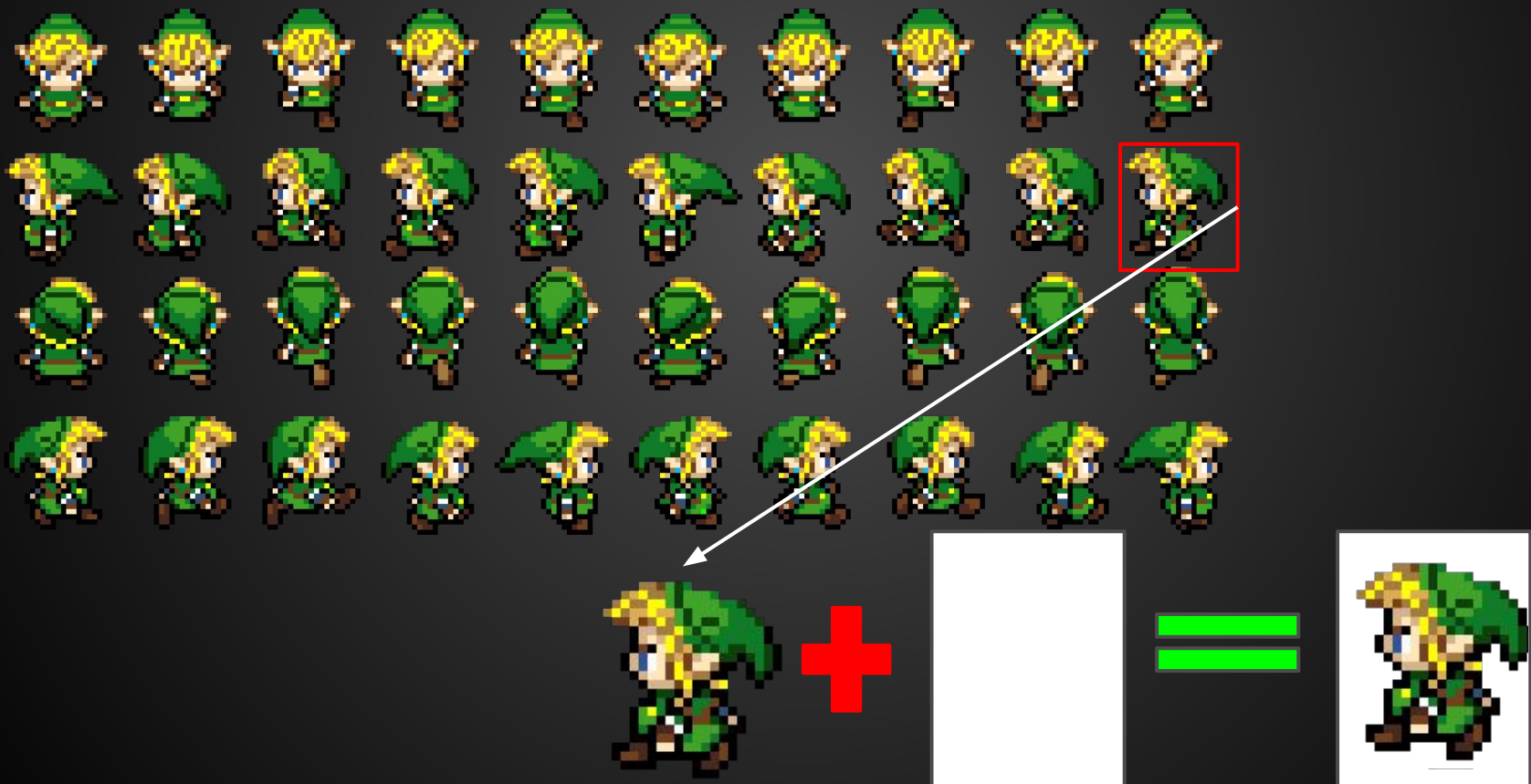
# Sprite - Animación



# Sprite - Animación



# Sprite - Animación





# Funciones avanzadas OpenGL

Es posible reemplazar todo o parte de una textura

- Ejemplo modelizar y visualizar un televisor en un escenario 3D



# Funciones avanzadas OpenGL

```
void glTexSubImage2D( GLenum target, GLint level, GLint x, GLint y, GLint w, GLint h, GLenum  
    format, GLenum type, GLvoid *texels );
```

target : GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D

level : 0 sin mip-mapping

x, y : coordenadas de la subregión a modificar

w, h : dimensiones de la subregión de la textura

format : GL\_RGB, GL\_RGBA, ...

type : type des datos de los texels (GL\_UNSIGNED\_BYTE)

texels : datos



# Funciones avanzadas OpenGL

- Manejo de texturas en la memoria
  - Las texturas son cargadas en la memoria del GPU, cuando no hay espacio OpenGL suprime las texturas
  - Verificar la validez de un índice de textura
    - GLboolean glIsTexture (GLuint texture);
    - Retorna GL\_TRUE - GL\_FALSE

# Funciones avanzadas OpenGL

- Manejo de texturas en la memoria
  - Verificación de la presencia de la textura en la memoria

`GLboolean glAreTexturesResident( GLsizei n, Gluint *texNums, GLboolean *residentes);`

- `n` : cantidad de texturas
- `texNums` : tabla de índices de texturas
- `Residentes` : tabla de booleanos (`TRUE == resident`)
- Si todas las texturas son residentes la función retorna `TRUE`, sino `FALSE`.

# Funciones avanzadas OpenGL

- Manejo de texturas en la memoria
  - Suprimir texturas
    - `glDeleteTextures(Gluint n, Gluint *tab_text);`

# Alpha blending

- La componente Alpha es la 4 componente que se agrega al modelo de color RGB. Mide la transparencia,
  - 0.0 corresponde a la transparencia total
  - 1.0 corresponde a un color totalmente opaco.
- Simulación de objetos translúcidos (agua, humo, ...), mezcla de imágenes, visualizar a través de objetos (camara video juegos).
- Hay dos comportamientos posibles
  - Transparencia (GL\_ALPHA\_TEST)
  - Translucidez (GL\_BLEND)

# Alpha blending

- Transparencia (GL\_ALPHA\_TEST)
- Los píxeles serán dibujado o no (comportamiento binario) en función al valor del componente alpha y del modo de la función glAlphaFunc().
- glAlphaFunc(mode, value)
  - Prueba la componente alpha de cada pixel a dibujar con el valor según el modo utilizado. Si el test se invalida el el pixel será transparente.

Modos :GL\_NEVER, GL\_LESS, GL\_EQUAL,  
GL\_LEQUAL, GL\_GREATER, GL\_NOTEQUAL,  
GL\_GEQUAL, GL\_ALWAYS

- GL\_ALWAYS todo los píxeles serán dibujados



# Alpha blending

- Ejemplo :

```
glAlphaFunc(GL_GREATER, 0);
```

```
glEnable(GL_ALPHA_TEST);
```

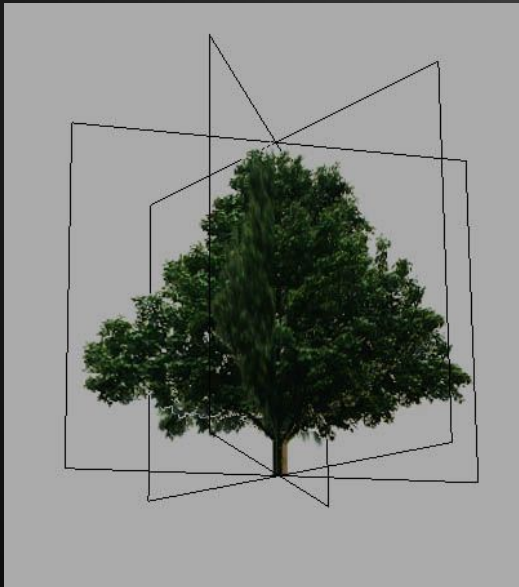
```
// Dibujar todas las texturas en RGBA utilizando el alpha test.
```

```
glDisable(GL_ALPHA_TEST);
```

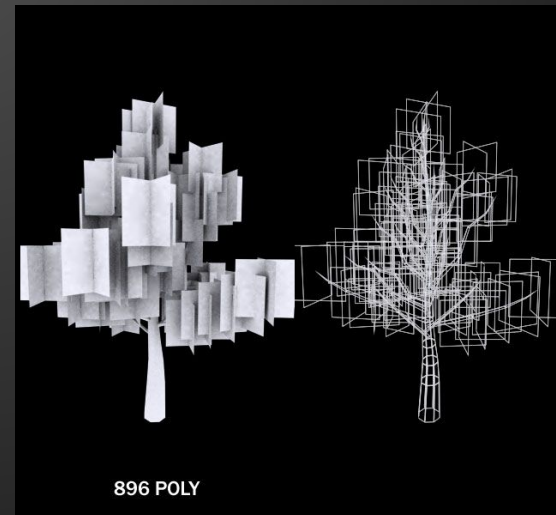
- los polígonos se dibujarán con “huecos” donde los texels de la textura no tienen su componente  $\alpha > 0$ .

# Alpha blending

- Observaciones :
- Técnicas muy utilizada para visualizar árboles en juegos o simuladores
- Al lugar de utilizar un árbol modelizado con miles de triángulos.



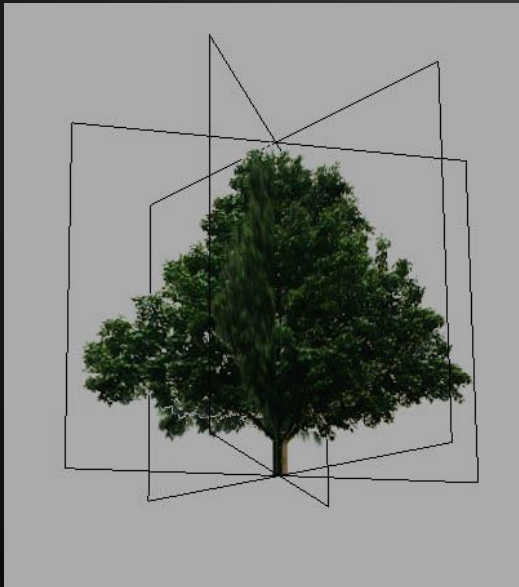
billboard tree



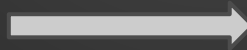
SpeedTree

# Alpha blending

- Observaciones :
- Técnicas que era muy utilizada para visualizar árboles en juegos o simuladores
- Al lugar de utilizar un árbol modelizado con miles de triángulos.



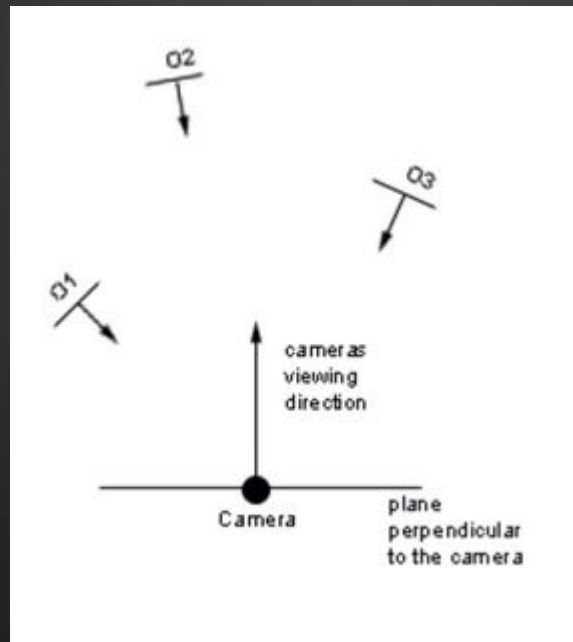
billboard tree



SpeedTree

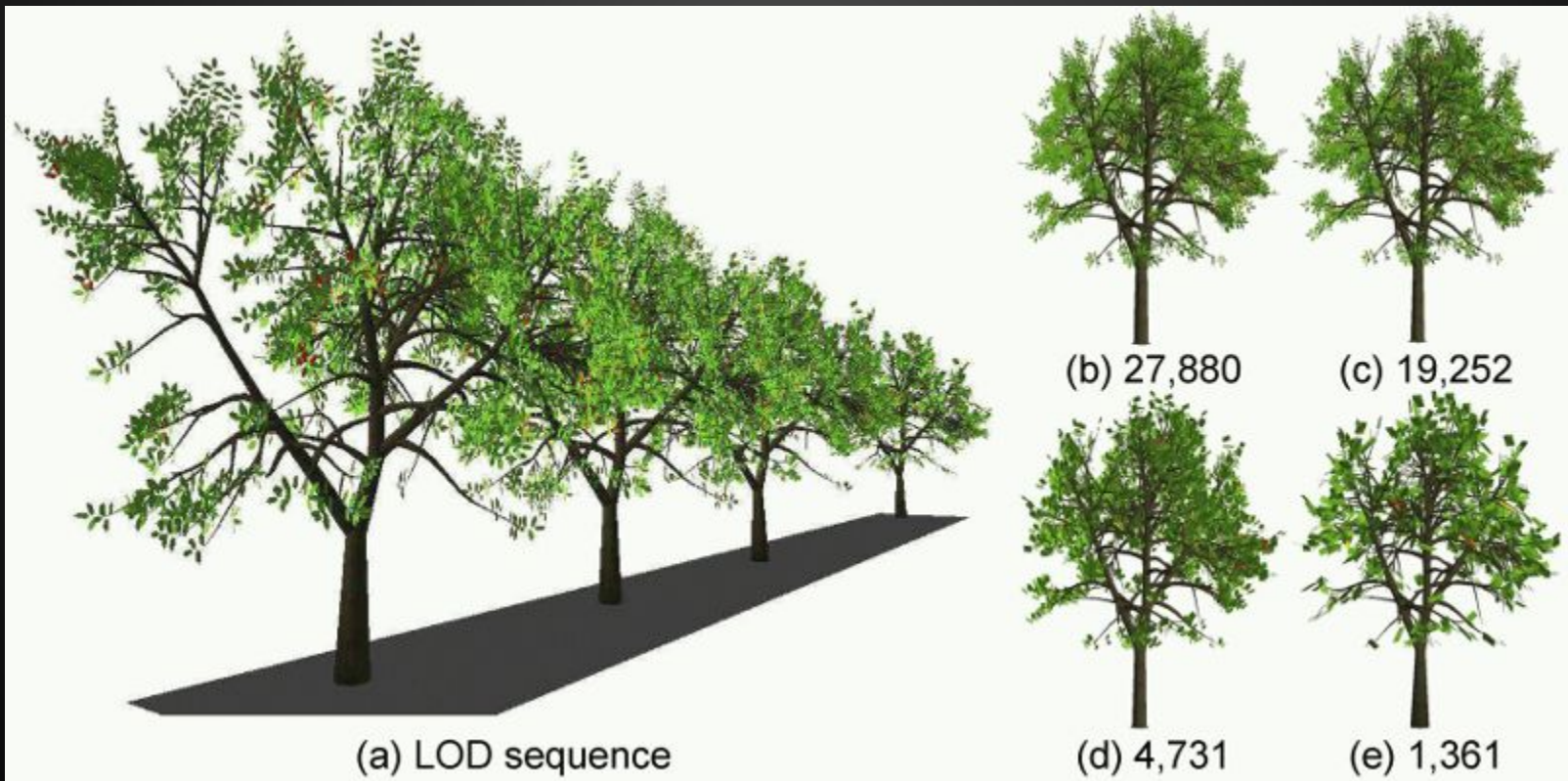
# Billboarding

- Es un cuadrilátero texturado (RGBA) que es siempre orientado hacia el jugador



# Billboarding

- Level of detail sobre árboles



# Alpha blending

- Translucidez
- El color de la textura es mezclado con lo demás de la escena en proporciones dadas por el valor de Alpha de cada texel en textura.
- La manera de mezclar los colores es dada por la función
  - `glBlendFunc(GL_SRC_ALPHA, mode);`
- El modo más utilizado es : `GL_ONE_MINUS_SRC_ALPHA`

# Alpha blending

- Ejemplo

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)  
;  
glEnable(GL_BLEND);  
// Dibujar todas las texturas en RGBA.  
glDisable(GL_BLEND);
```

Los polígonos serán más o menos translúcidos.



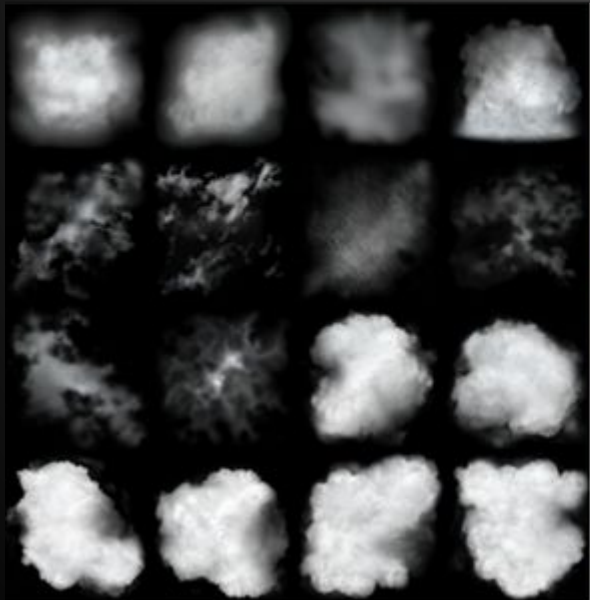
# Alpha blending

Utilizado para efectos de humo



# Alpha blending

En flight simulator de microsoft las nubes son obtenidas a partir de cientos de billboards con texturas y tamaños diferentes. Este color es modulado : más oscuro en la base de los billboards y más claro arriba para simular la atenuación de la luz pasando a través de las nubes.



# Multitexturing

Vimos cómo mapear **una sola** textura sobre un polígono. Las tarjetas gráficas permiten mapear **varias** texturas sobre el mismo polígono combinando estas texturas según varias situaciones.

Para realizar estas técnicas se utilizan shaders.

# Multitexturing

Lightmap : Se mezcla la textura del color con una textura que almacena una iluminación precalculada. Técnica, muy utilizada desde el juego quake.





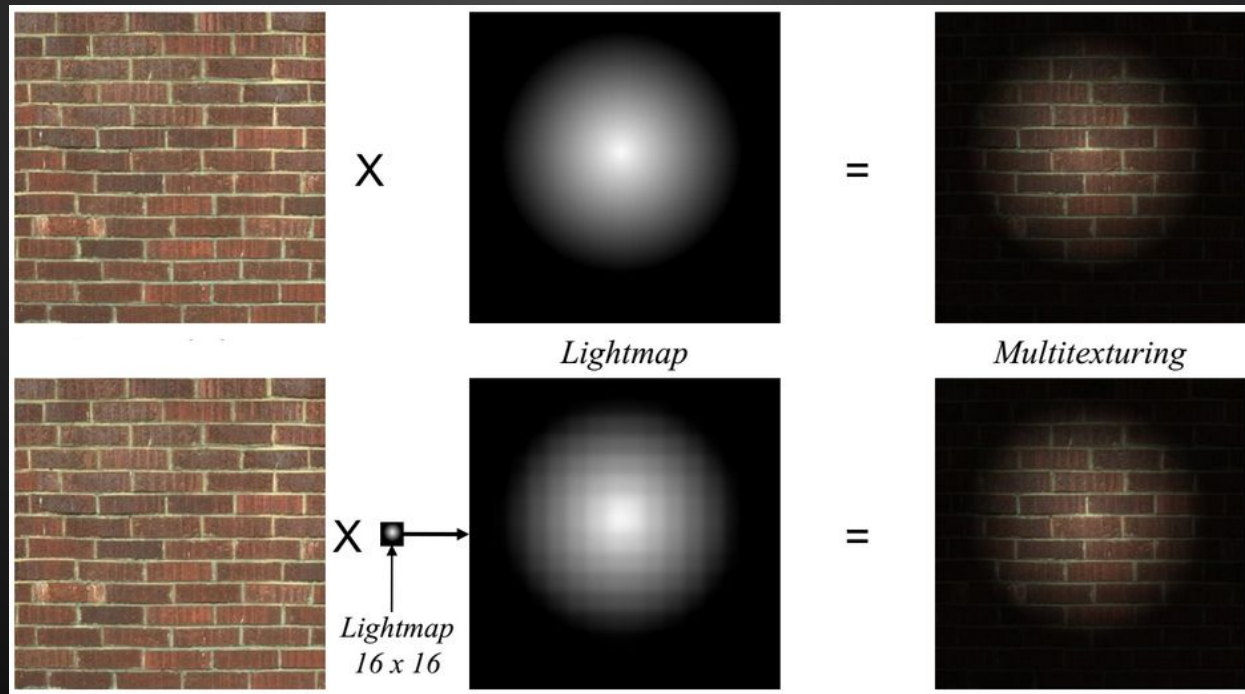
# Multitexturing

Lightmap : Se mezcla la textura del color con una textura que almacena una iluminación precalculada. Técnica, muy utilizada desde el juego quake.



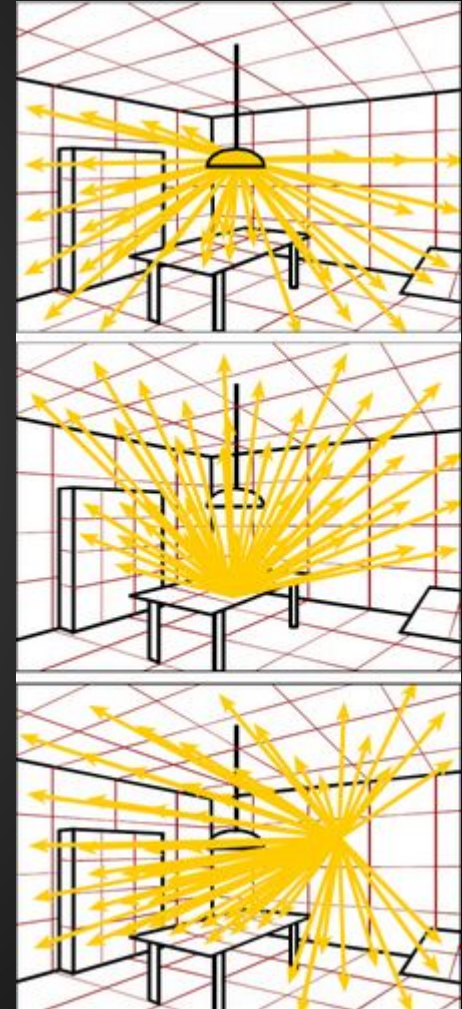
# Multitexturing

- Las texturas de lightmap contienen solamente una componente de luminosidad.
- No necesitan de ser de resolución alta.
- No necesitamos tener varias texturas original según la iluminación.



# Multitexturing

Las lightmap son generalmente calculados gracias a la radiosidad (global iluminación). La idea es de descomponer las superficies en cuadrados (“patches”). Algunos de estos cuadrados corresponden a las fuentes de luz. Calculamos después los intercambios de luz entre todos estos cuadrados.

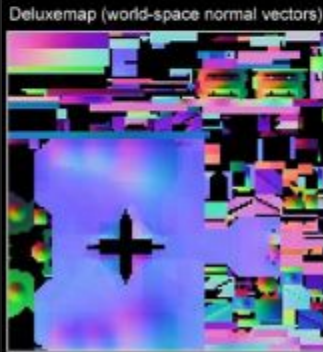




# Multitexturing



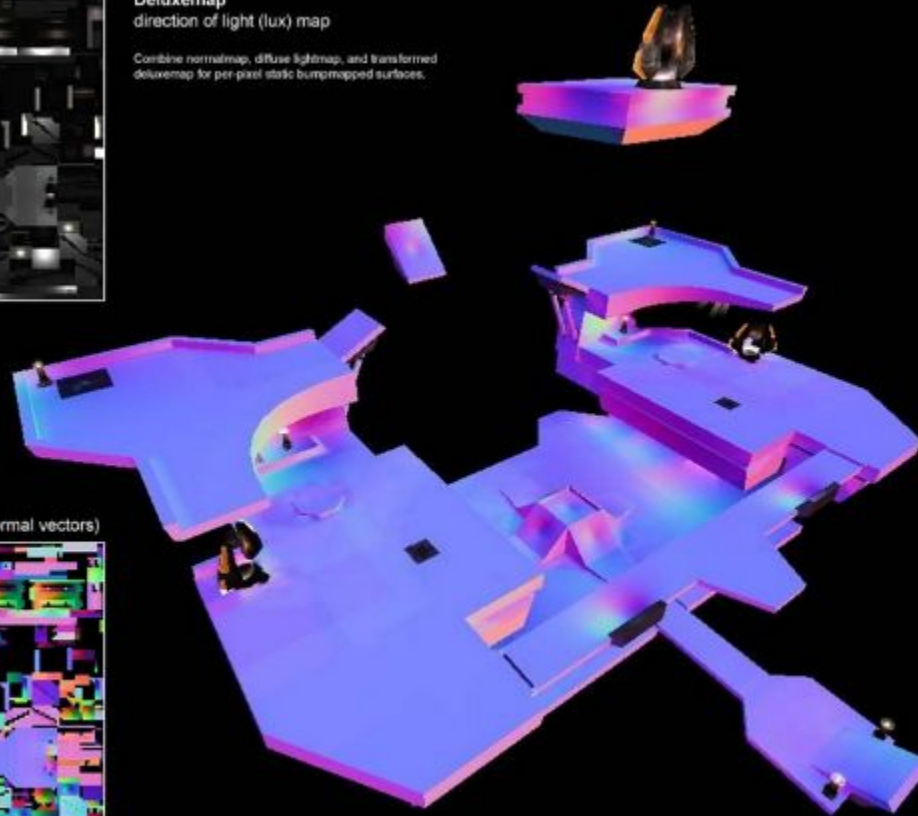
Lightmap (RGB)



Deluxemap (world-space normal vectors)

**Deluxemap**  
direction of light (lux) map

Combine normalmap, diffuse lightmap, and transformed deluxemap for per-pixel static bumpmapped surfaces.



# Multitexturing

Tamaños de lightmap exagerados provocan efectos de aliasing



# Environment Mapping

Permite simular superficies reflejantes (chrome, metal, ...) gracias a un mapeo de textura.



# Environment Mapping

Las coordenadas de texturas del objeto reflexivo son calculadas representando el entorno según la posición del observador.

- Varios métodos integrados en las tarjetas gráficas
  - sphere mapping
  - cube mapping
  - ...

# Environment Mapping

Sphere mapping :

Utilizando una textura representando una esfera perfectamente reflexiva. Para cada vértice calculamos coordenadas de textura (xt, yt ) en función a la normal (x,y,z) en este vértice.



Environment Texture



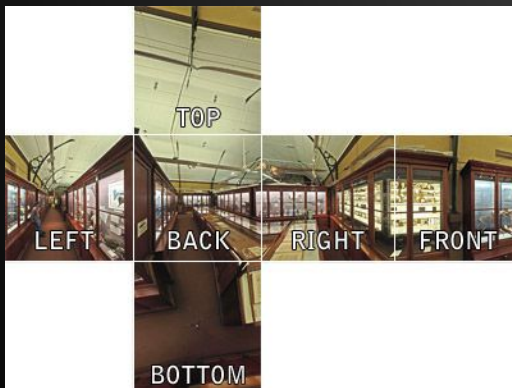
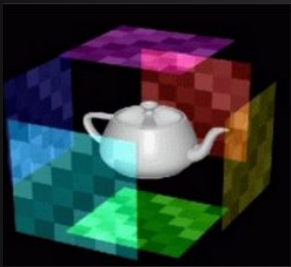
# Environment Mapping

```
// Un sola vez :  
glTexGenfv(GL_S, GL_SPHERE_MAP, 0);  
glTexGenfv(GL_T, GL_SPHERE_MAP, 0);  
// Activar la generación de coordenadas de textura  
glEnable(GL_TEXTURE_GEN_S);  
glEnable(GL_TEXTURE_GEN_T);  
// Visualizar el objeto reflexivo  
// ...  
// Desactivar la generación de coordenadas de textura  
glDisable(GL_TEXTURE_GEN_S);  
glDisable(GL_TEXTURE_GEN_T);
```



# Environment Mapping

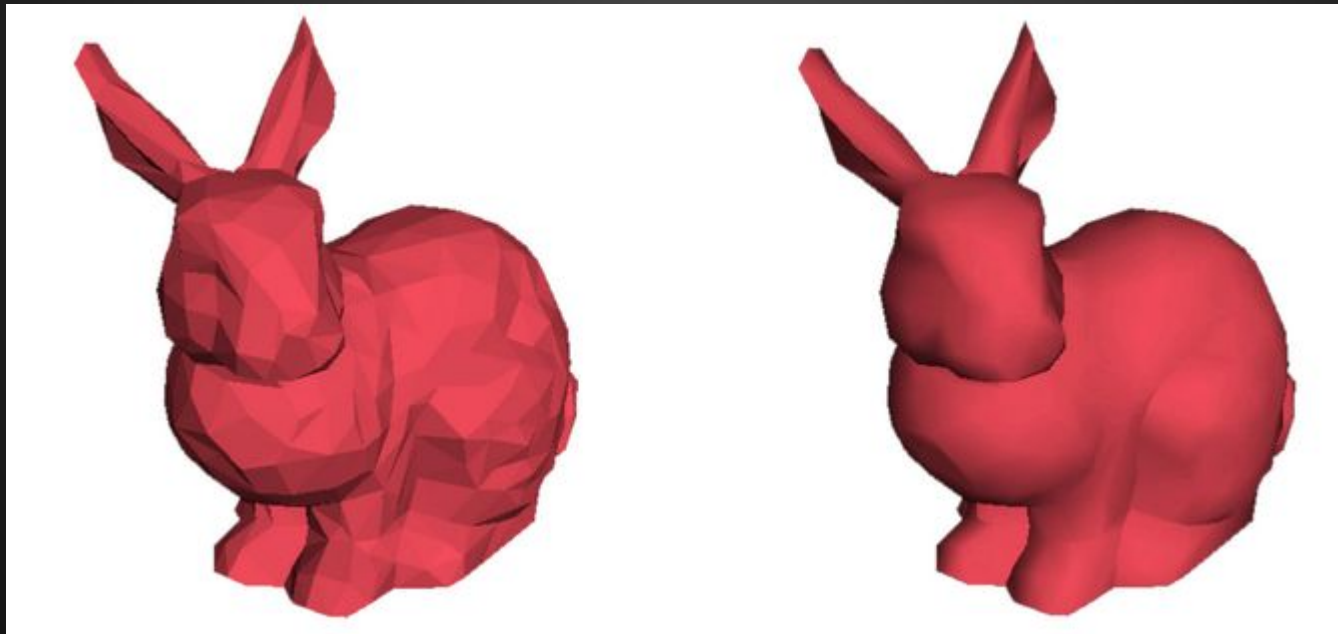
Cube mapping : el objeto a texturar se encuentra en un cubo en cual las seis caras representan el entorno del objeto. (Da mejores resultados que el sphere mapping pero más costoso - 6 imágenes)





# Bump mapping

Alisamiento de Gouraud : calcular la iluminación por cada vértice gracias a su normal (promedio de las normales de los triángulos vecinos) y interpolar los colores sobre toda la cara. Obtenemos un superficie lisa con variaciones suaves de las normales.



Flat shading

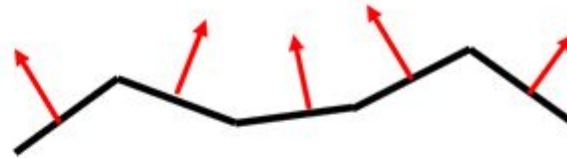
Alisamiento de Gouraud

# Bump mapping

El bump mapping se basa en la idea de modificar la normales a nivel de píxeles para modificar la iluminación de la superficie



Superficie simple

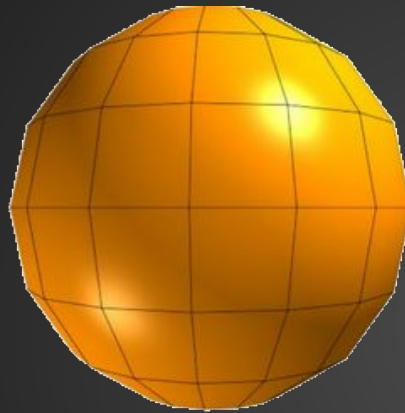


Geometría a simular

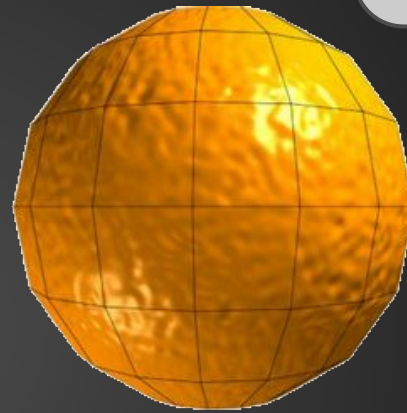


Superficie con bump mapping

# Bump mapping



Sin bump mapping



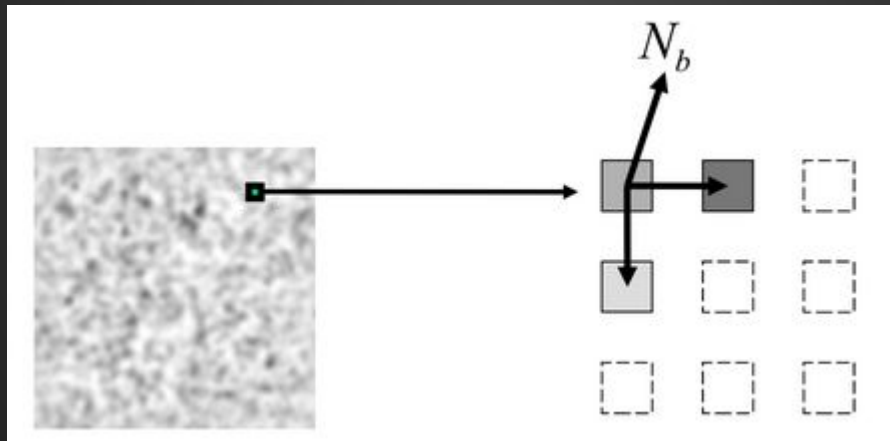
con bump mapping

Perturbación de las normales con una textura de bump map

Los dos objetos tienen exactamente la misma geometría, la misma cantidad de polígonos. La iluminación es lo único que varía.

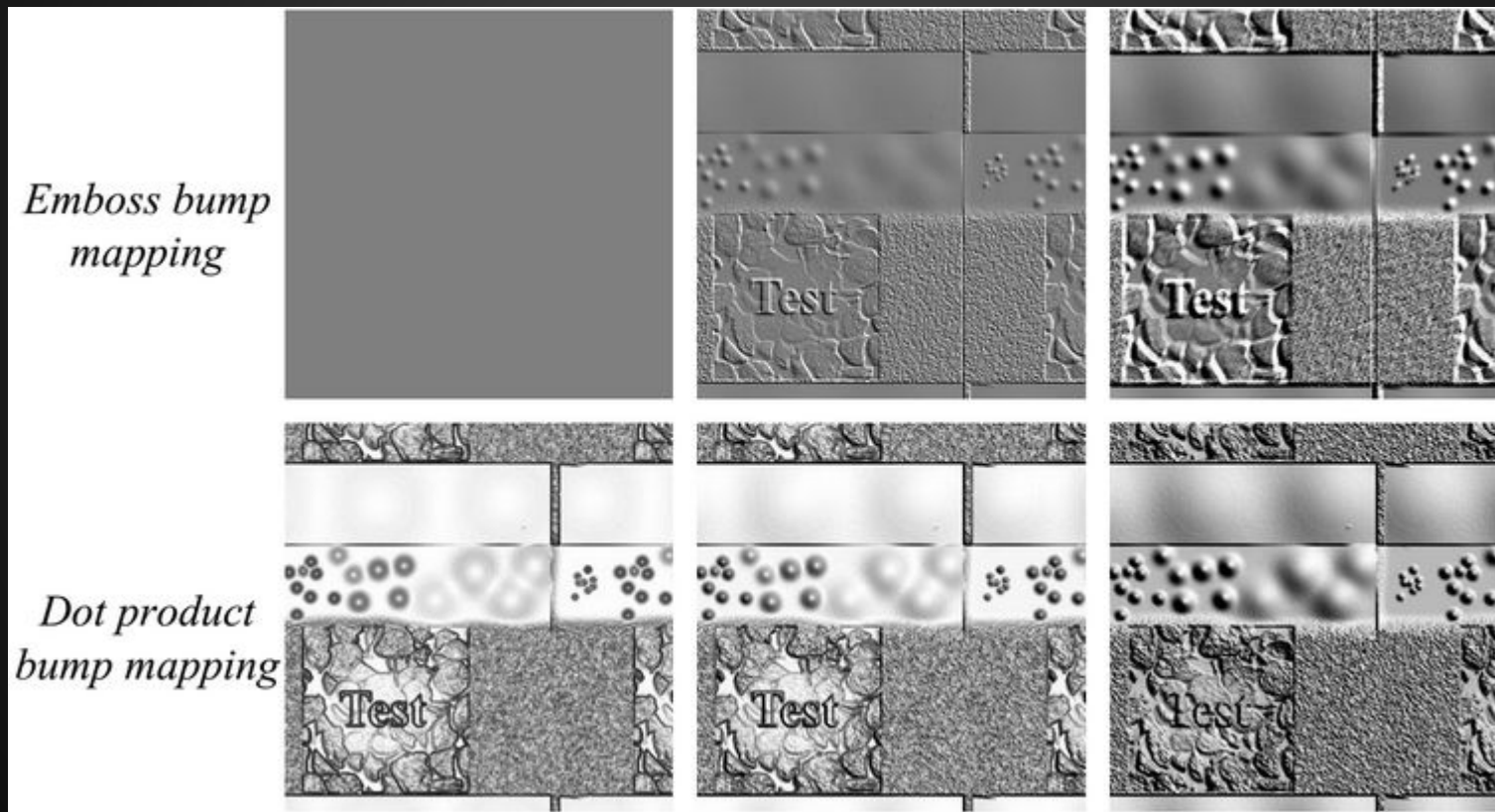
# Bump mapping

Por cada pixel a visualizar, queremos modificar su normal  $N$ . Utilizamos una imagen en niveles de grises para calcular la nueva normal que será utilizada para perturbar la normal  $N$ . La imagen en niveles de grises es un mapa de alturas. A partir de 3 pixeles determinamos la normal  $N$



# Bump mapping

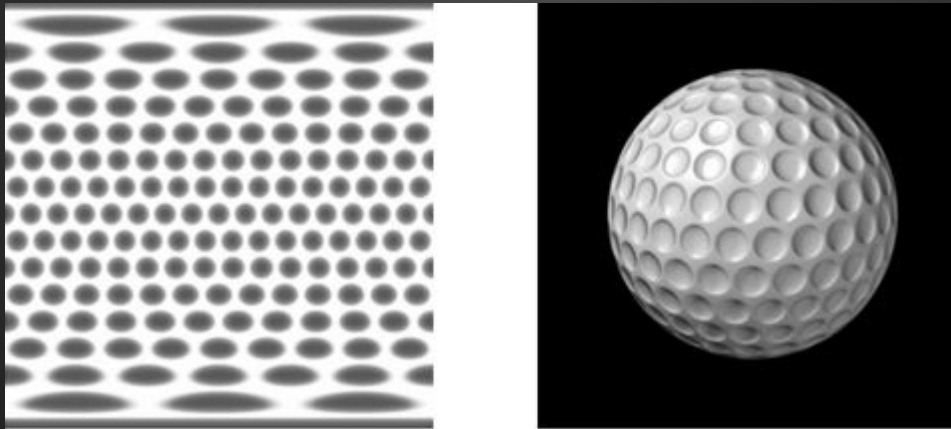
Varias tecnicas de bump mapping aparecieron en las tarjetas grafica mas o menos rigurosas y costosas en tiempo de cálculo.



Ahora todo se hace por shaders

# Bump mapping

Varias técnicas de bump mapping aparecieron en las tarjetas gráficas más o menos rigurosas y costosas en tiempo de cálculo.



<https://www.youtube.com/watch?v=ODivkMAfZ18>

Ahora todo se hace por shaders

# Normal mapping

Técnica de almacenamiento de las normales de una superficie en una textura utilizada según el método de bumping. La diferencia con el bump-mapping es que son imágenes en 24 bits que son utilizadas (3 X mas espacio en memoria) : los componentes RGB de los pixeles sirven a codificar las coordenadas (x,y,z) de la normal.



Geometría 3D



Proyección de la geometría

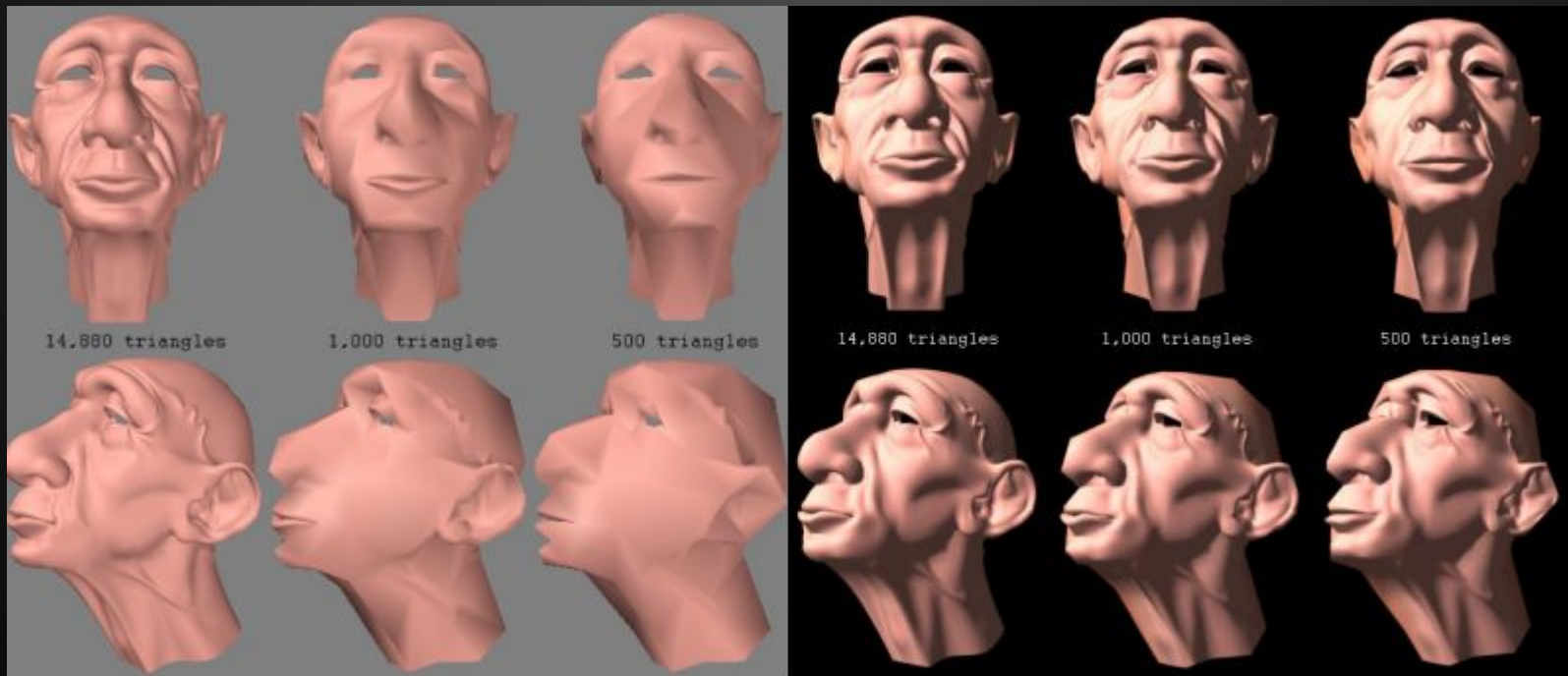


Normal map



# Normal mapping

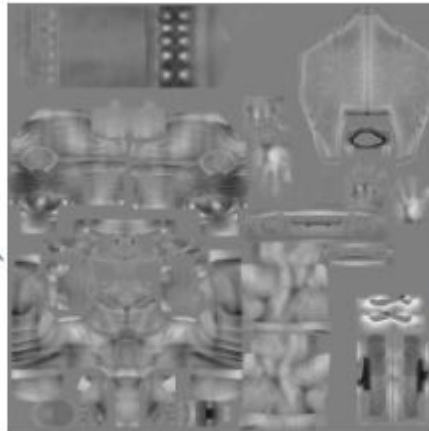
A partir de un modelo 3D muy detallado, calcular normales y almacenarlas en una textura normal map.  
Utilizar esta normal map sobre un modelo 3D low-poly.  
Técnica utilizada en casi todo los video juegos.



<https://www.youtube.com/watch?v=C2uFD2R1IN0>

# Displacement mapping

Aumenta la complejidad (nivel de detalles) de una superficie generando polígonos a partir de texturas. Al contrario del bump-mapping, la superficie es realmente modificada.

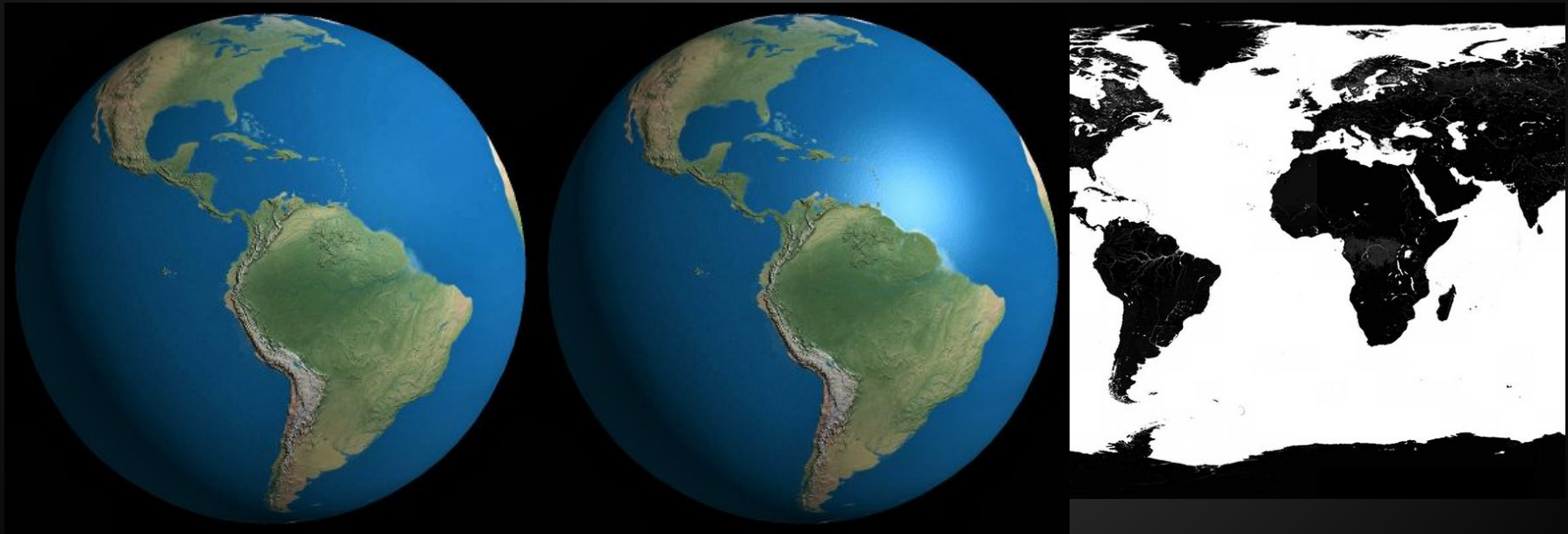


*Displacement map*



# specular mapping

- Se almacena en una textura la intensidad de luz especular reflejada por pixel
  - Negro = no refleja ninguna luz especular
  - Blanco = refleja toda la luz especular

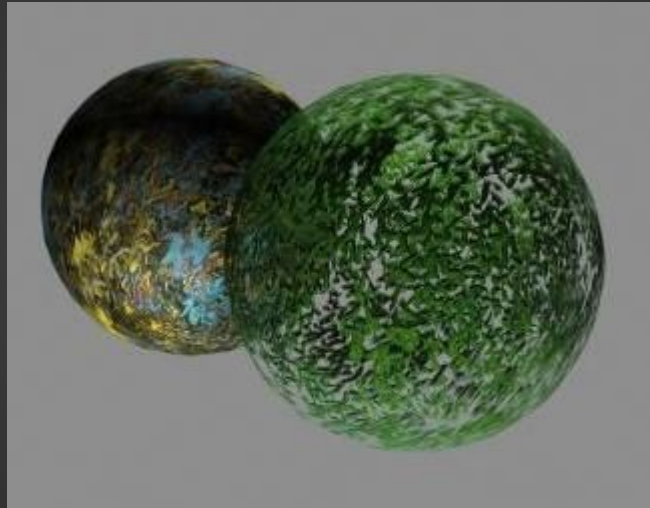


# gloss mapping

- Se almacena en una textura el coeficiente especular por pixel
  - Negro = El coeficiente especular es bajo (mate)
  - Blanco = El coeficiente especular es alto (brillante)

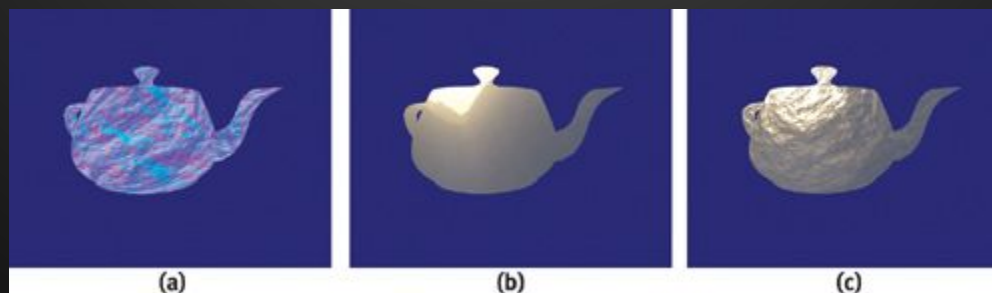
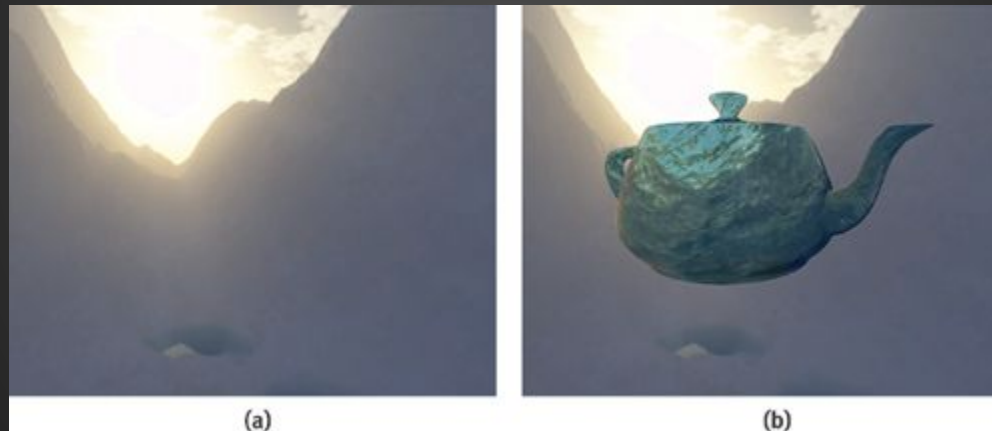
# opacity mapping

- Se almacena en una textura el canal alpha de la textura al lugar de tener una textura RGBA.



# Reflection mapping

- Render to texture de la escena sin el objeto refractivo.
- aplicar la textura de reflexión que almacena los vectores que desviarán la luz a través del objeto
- aplicar la textura del fondo de la escena al objeto refractivo.
- distorsionar la textura del fondo gracias a los vectores del refraction map





# Ambient occlusion

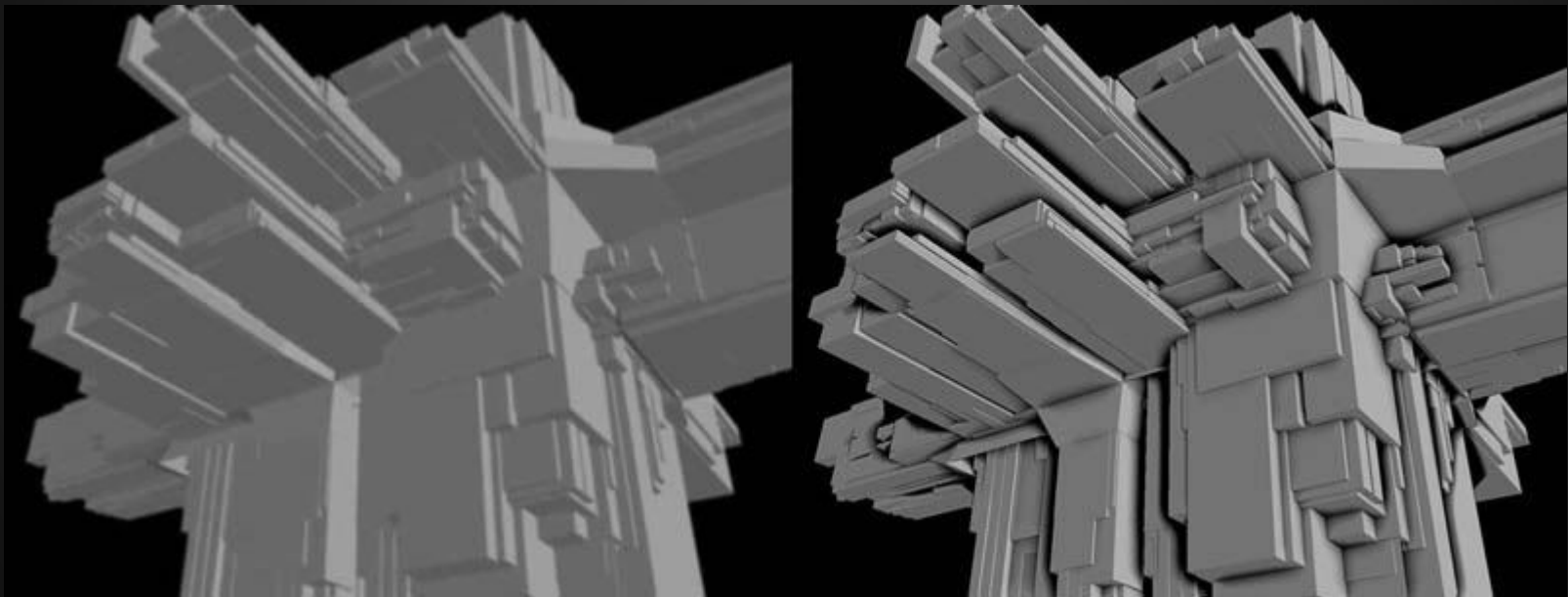
- El ambient occlusion permite simular el acceso difícil a la luz en algunas partes de un objeto, generalmente las esquinas cóncavas. Son texturas precalculadas, el método es parecido al lightmapping.





# Ambient occlusion

- El ambient occlusion permite simular el acceso difícil a la luz en algunas partes de un objeto, generalmente las esquinas cóncavas. Son texturas precalculadas, el método es parecido al lightmapping.



<http://www.geforce.com/whats-new/guides/watch-dogs-graphics-performance-and-tweaking-guide#ambient-occlusion>

# Como mandar varias texturas

- Cargar varias texturas

```
GLuint DiffuseTexture = loadTexture("diffuse.png");
```

```
GLuint NormalTexture = loadTexture("normal.png");
```

- Recuperar los textureSampler

```
GLuint DiffuseTextureID = glGetUniformLocation(programID, "DiffuseTextureSampler");
```

```
GLuint NormalTextureID = glGetUniformLocation(programID, "NormalTextureSampler");
```

- Bucle de rendering

```
glActiveTexture(GL_TEXTURE0);
```

```
glBindTexture(GL_TEXTURE_2D, DiffuseTexture);
```

```
glUniform1i(DiffuseTextureID, 0);
```

```
glActiveTexture(GL_TEXTURE1);
```

```
glBindTexture(GL_TEXTURE_2D, NormalTexture);
```

```
glUniform1i(NormalTextureID, 1);
```