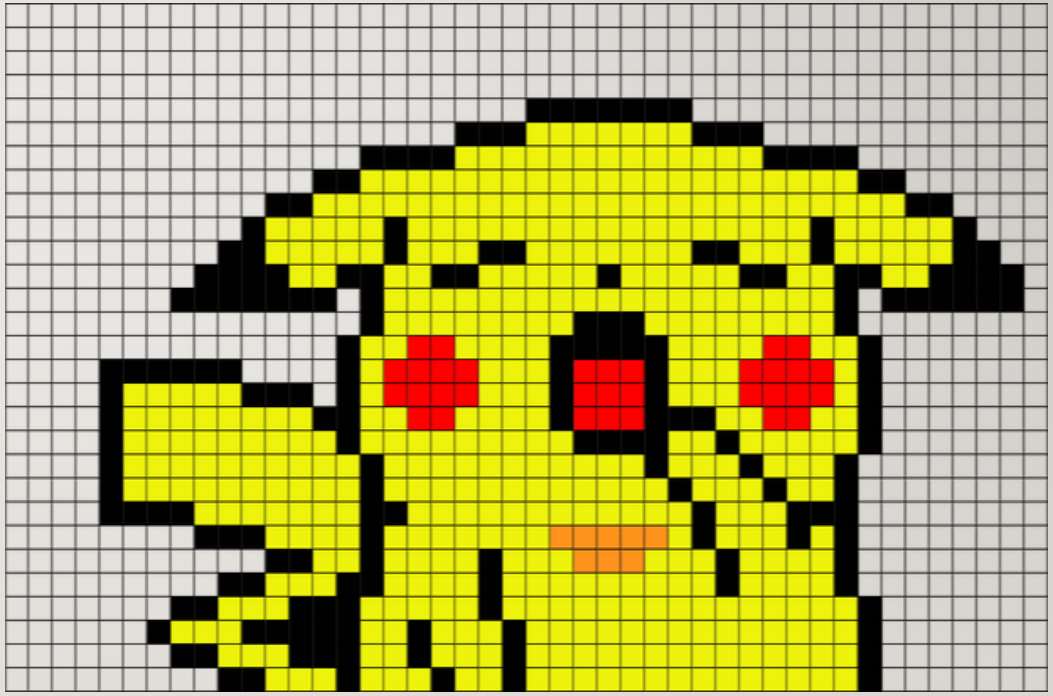
TEXTURIZADO

TEXTURA, ES UN ARREGLO DE DATOS QUE CONTIENE INFORMACIÓN DE COLOR Y TRANSPARENCIA.

ACTUALMENTE, TAMBIÉN PUEDE CONTENER INFORMACIÓN DE: NORMALES, PROFUNDIDADES, SOMBRAS, ENTRE OTROS.

TEXEL

- Los elementos que forman parte de una textura reciben el nombre de **Texel**. (**Tex**ture **El**ement o **Tex**ture Pix**el**)



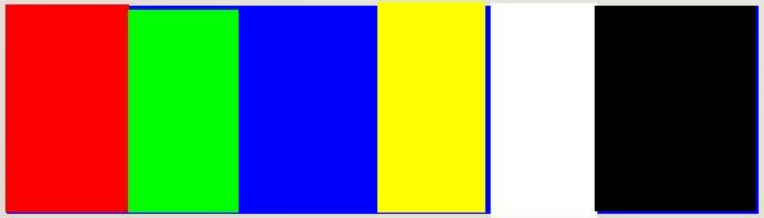
TIPOS DE TEXTURAS

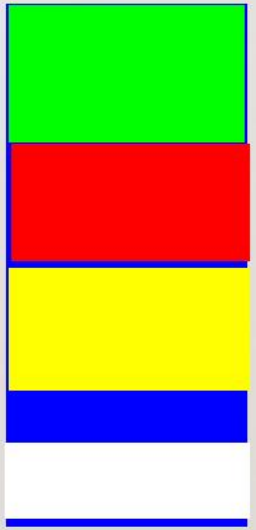
Existen dos clasificaciones para determinar los tipos de texturas:

- Por el tamaño de las texturas.
- Por la forma en que son "generadas" las texturas.

TEXTURA 1D

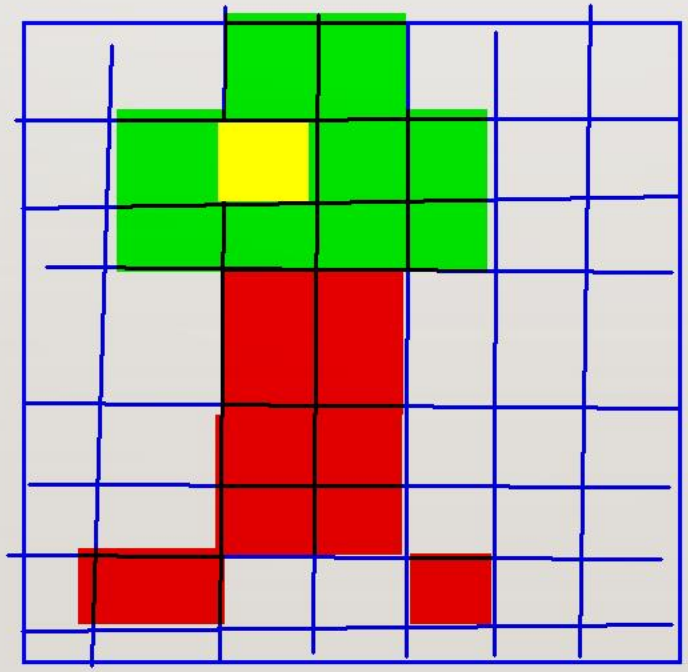
- Un texel de Ancho o un texel de alto



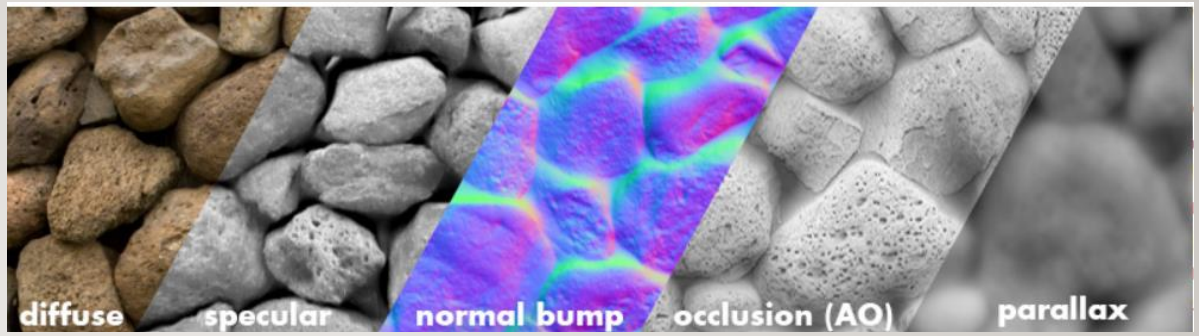


TEXTURA 2D

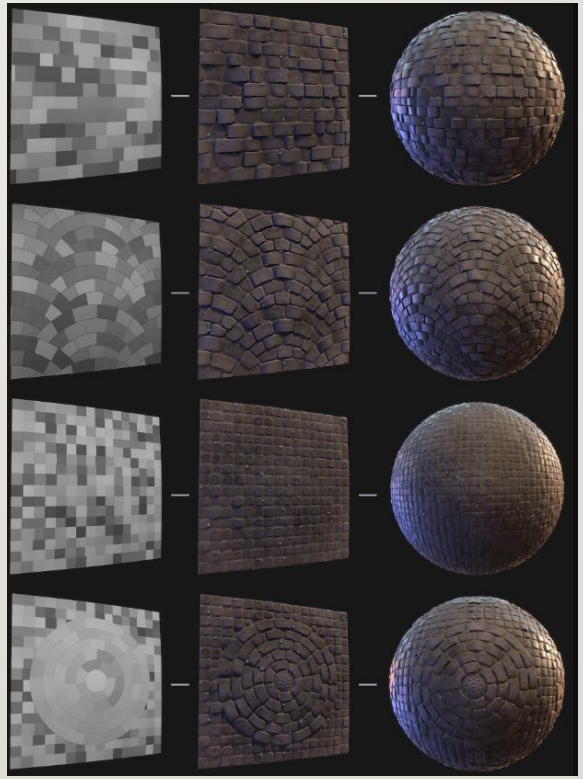
- Más de un texel de ancho y de alto

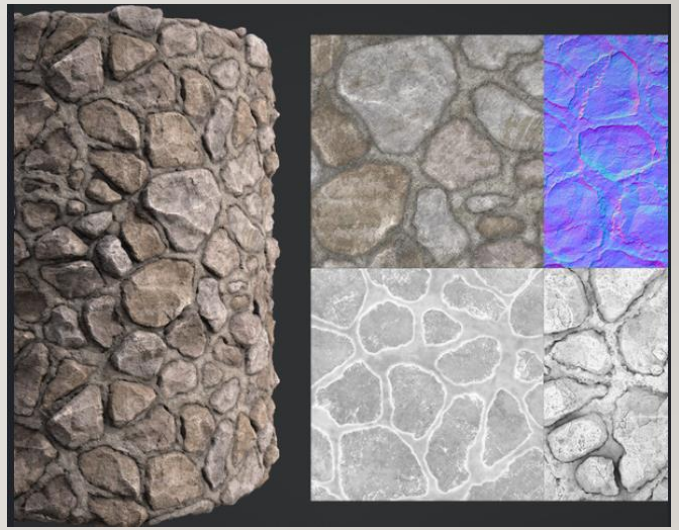


TEXTURAS 3D



- Manejan volumen, se usa la información de más de una textura 2D





TEXTURAS POR TIPO DE GENERACIÓN

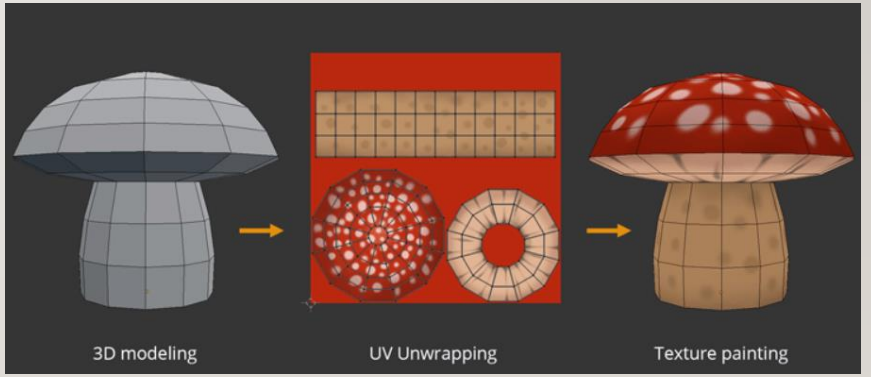
- Archivos almacenados en memoria (Imágenes)

TEXTURAS POR TIPO DE GENERACIÓN

- Procedurales:
- Se generan por medio de algoritmos o funciones que nos permiten obtener patrones que corresponden a texels. Generalmente a los algoritmos se les agrega ruido.

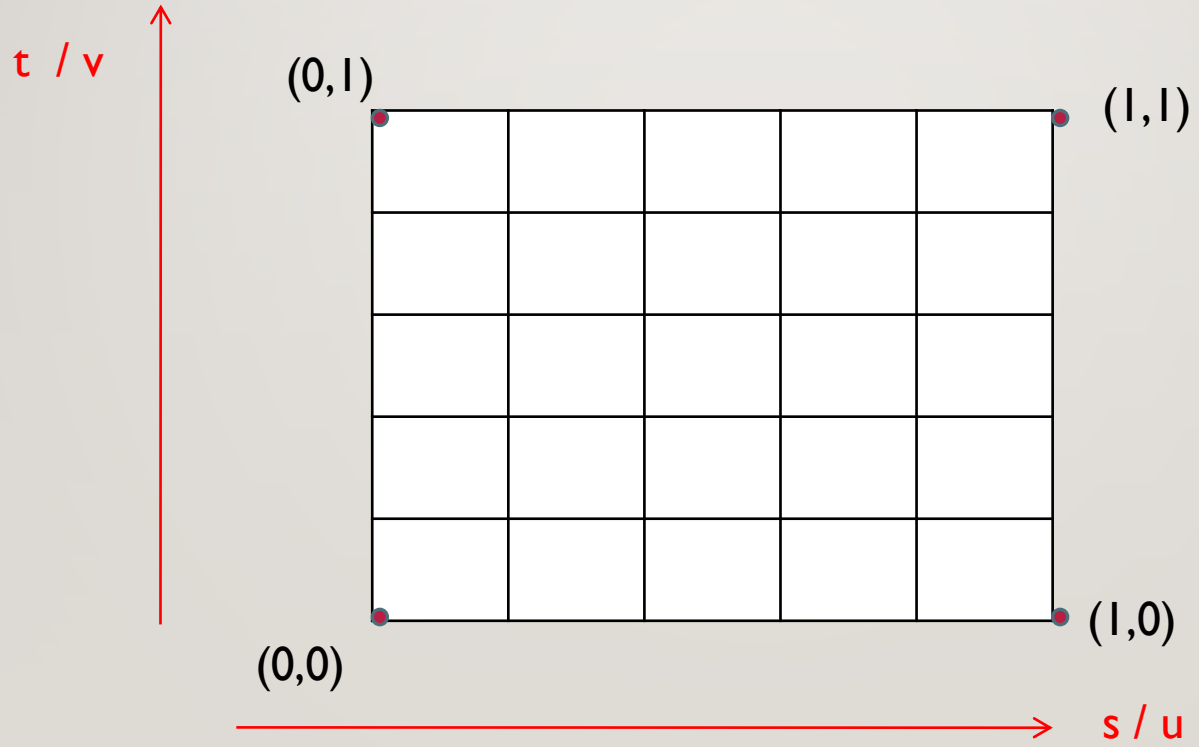
TEXTURE MAPPING

- Regla de correspondencia para aplicar una Textura a una superficie.
- Aplicar a un objeto una textura, hacemos relación entre el espacio de coordenadas de texturizado (S,T) o (U,V) sobre coordenadas espaciales (X,Y) (X,Y,Z).
- A cada pixel de la superficie se le asigna el correspondiente texel de la textura



ESPACIO DE TEXTURIZADO

- Espacio de Coordenadas (U,V) o (S,T) rango de [0,1]



TEXTURAS EN OPENGL

En OpenGL se deben de seguir varios pasos para utilizar texturizado.

El comando que recibe muchos parámetros de configuración es glTexParameter*

glTexParameteri(GL_TEXTURE_TIPO, PARÁMETRO,VALOR);

TEXTURE WRAPPING

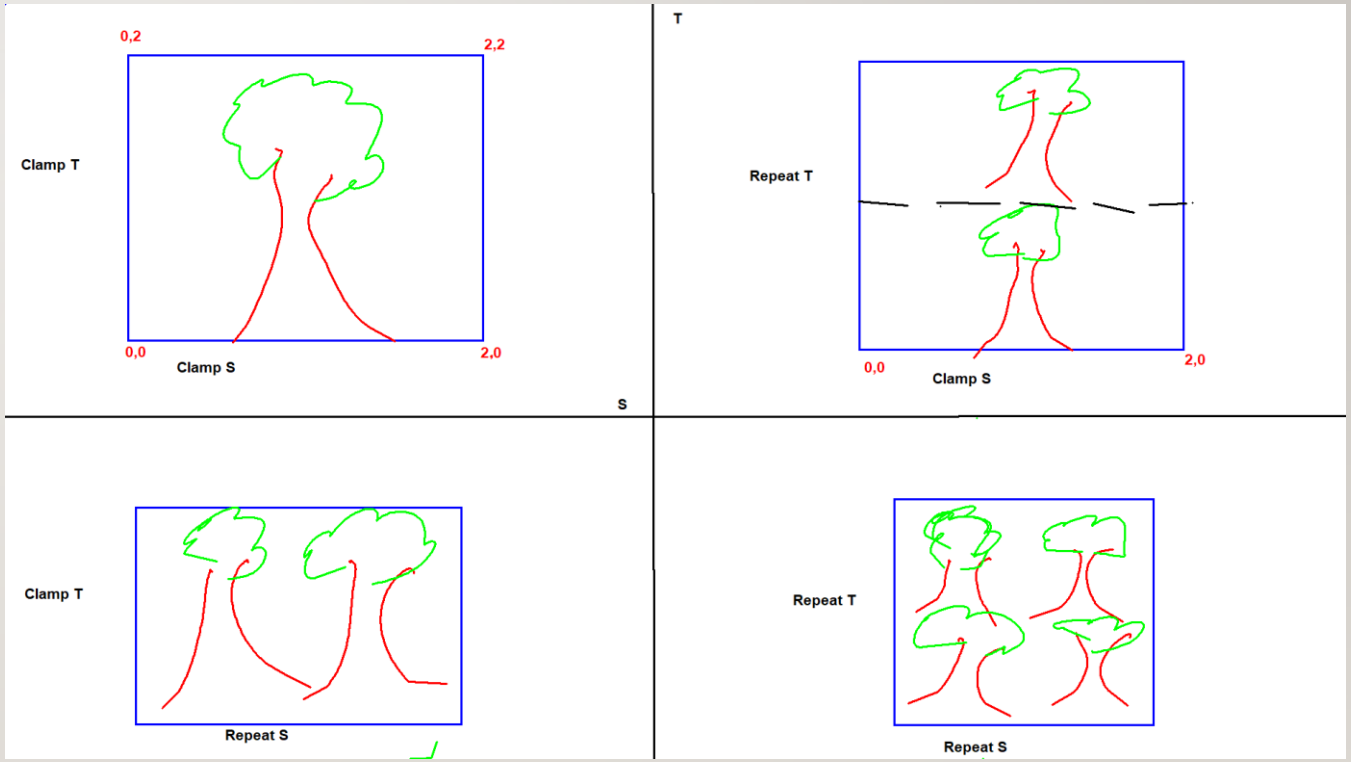
- El wrapping (envolver) es indicar la textura como se comportará sobre la superficie del objeto con coordenadas fuera del rango de (0,1)

•GL_REPEAT
•GL_MIRRORED_REPEAT
•GL_CLAMP

TEXTURE WRAPPING

- El wrapping (envolver) es indicar la textura como se comportará sobre la superficie del objeto con coordenadas fuera del rango de (0,1)

•GL_REPEAT
•GL_MIRRORED_REPEAT
•GL_CLAMP



FILTROS

linear

bilinear

trilinear

- Los filtros son algoritmos que se aplican a las texturas para que al aplicarlas a los objetos se obtenga un mejor resultado si se redimensionan.
- Los filtros más utilizados son:

❑ GL_NEAREST. — Estilo visual pixeleado o de 8 bits. A cada pixel le corresponde el texel más cercano

❑ GL_LINEAR. — A cada pixel le corresponde el promedio de los 4 texel más cercanos

❑ GL_NEAREST_MIPMAP_NEAREST. — Al mipmap más cercano aplicale el filtro NEAREST

❑ GL_LINEAR_MIPMAP_NEAREST. — Al mipmap más cercano aplícale filtro LINEAR

❑ GL_NEAREST_MIPMAP_LINEAR. — A los dos mipmap más cercanos, aplicales filtro NEAREST y promedia los resultados

❑ GL_LINEAR_MIPMAP_LINEAR. — A los dos mipmaps más cercanos, aplícales flitro LINEAR y promedia los resultados

❑ Anisotripic Filtering (Filtro Anisotrópico).

MIP MAPS

- Son secuencias de las texturas precalculadas, dichas secuencias son para menor resolución de la textura origen y son progresivos.
- MIP viene del latín: "Multum in parvo" que significa "mucho en poco"
- glGenerateMipmap(GL_TEXTURE_2D);

MIP MAPS

- Son secuencias de las texturas precalculadas, dichas secuencias son para menor resolución de la textura origen y son progresivos.
- MIP viene del latín: "Multum in parvo" que significa "mucho en poco"
- glGenerateMipmap(GL_TEXTURE_2D);

ANISOTROPICO

- el filtro anisotrópico se aplica en superficies oblicuas con respecto a la cámara. También se puede aplicar si la textura tiene información de ángulo al ser generada

PASOS A SEGUIR EN OPENGL PARA CREAR TEXTURAS 2D

0.- Este es un paso que no nos requiere OpenGL, pero que debemos de tomar en cuenta nosotros: Tener una imagen con las características adecuadas para que se cree la textura correctamente

- Formato de Color de la imagen
- Extensión de la imagen
- Dimensión de la imagen

1. Tener un cargador de imágenes : stb_image, SOIL, GL image, etc..
   - *#include "texture.h"*
   - El cargador lee la imagen, verifica la información de la imagen, la información la almacena en un arreglo de datos
     - t_Ajedrez1.LoadTGA("02.tga");

PASOS A SEGUIR EN OPENGL PARA CREAR TEXTURAS 2D

2. Crear una textura
   - Para crear una textura leemos el arreglo de datos creado con el cargador de imágenes, generamos la textura y ligamos la textura
     - *t_Ajedrez1.BuildGLTexture();*
     - *glGenTextures(1,&GLindex);*
     - *imageData=new unsigned char [imageSize];*
   - Se dan los parámetros en el eje S, eje T, filtros, formato de color e información, se generan mipmaps
     - *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_MIN_FILTER, GL_LINEAR);*
     - *glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);*
     - glTexImage2D(GL_TEXTURE_2D, 0, color_type, width, height,0, color_type, GL_UNSIGNED_BYTE, imageData);
     - *glGenerateMipmap(GL_TEXTURE_2D);*

PASOS A SEGUIR EN OPENGL PARA CREAR TEXTURAS 2D

3. Se asigna la textura (Bind) : *DadoTexture.UseTexture();*
   - *Se liga la textura al objeto*
     - glBindTexture(GL_TEXTURE_2D, textureID);
4. Se libera la imagen
   - t_Ajedrez1.ReleaseImage();//para liberar la información de la imagen

PASOS A SEGUIR EN OPENGL PARA CREAR TEXTURAS 2D

- 5.- Se mapea la imagen
- glBindTexture(GL_TEXTURE_2D, textura1); // choose the texture to use.
- glBegin(GL_POLYGON); //Top
  - glNormal3f(0.0f, 1.0f, 0.0f);
  - glTexCoord2f(1.0, 0.0f); glVertex3fv(vertice[2]);
  - glTexCoord2f(1.0, 1.0f); glVertex3fv(vertice[3]);
  - glTexCoord2f(0.0, 1.0f); glVertex3fv(vertice[4]);
  - glTexCoord2f(0.0f, 0.0f); glVertex3fv(vertice[7]);
  - glTexCoord2f(1.0, 0.0f); glVertex3fv(vertice[8]);
  - glTexCoord2f(1.0, 1.0f); glVertex3fv(vertice[11]);
- glEnd();

PASOS A SEGUIR EN OPENGL PARA CREAR TEXTURAS 2D

- Si se tiene Iluminación, se debe desactivar la Iluminación antes de renderizar al objeto con textura y activarla después de renderizar al objeto
- glPushMatrix();
  - glColor3f(1.0, 1.0, 1.0);
  - glScalef(1.0,1.0,1.0);
  - glDisable(GL_LIGHTING);
  - prisma(t_Ajedrez2.GLindex, t_metal01.GLindex);
  - glEnable(GL_LIGHTING);
- glPopMatrix();