# Recordatorio

Estuuvimos hablando del **Content** **Pipeline** y los distintos tipos de recursos.

**Texturas**:

* Width y heigt iguales y potencia de 2.
* Formato **texel**
* **Numero** de **canales** (normalmente 4)
* **Numero** de **bits** (8 normalmente)
* **Compresión**: las tarjetas graficas usaban un factor mas o menos de 4 a 8.

# MIPMAPS

Teniendo una textura de NxN, el Hardware decide tener la textura de N/2,N/2

Y el primer pixel nuevo sale de los 4 primeros pixeles de la primera textura

Pizarrón blanco con texto en letras negras sobre fondo blanco

Descripción generada automáticamente con confianza mediaPizarrón blanco con letras negras

Descripción generada automáticamente con confianza media

Y cuando tiene la primera, lo vuelve a hacer hasta tener una textura de 1x1

Si teníamos 4MB, ahora ocuparemos 5MB o así. ¿Pero **por qué** hacemos esto entonces?

Yo tengo una textura de 512x512 de un suelo.

Imagen que contiene texto, pizarrón, hombre, surfeando

Descripción generada automáticamente

Si tengo una textura que es blanco, negro, blanco, negro tengo un problema. Y entonces para las baldosas del final podría usar cualquier cosa para un pixel. ALIASING.

Las texturas de la derecha de la cadena de Mipmaps tendrán menos frecuencia que los de la izquierda, y se verán la media de los pixeles.

A efectos de cache también es muy bueno pero ocupa un poco más.

Por tanto: **+RENDIMIENTO, pero +TAMAÑO**

El formato **DDS** es un contenedor de esta información que además permite guardar pregenerados los mipmaps. Se pueden ver los mipmaps calculados y modificarlos.

Ejemeplo SIN MIPMAP

En el fondo a veces hay rojo y a veces negro 🡪 ALIASING y nadie quiere ver estos dientes de sierra en nuestro juego

Patrón de fondo

Descripción generada automáticamente con confianza media

Aquí es discreto (el que esté mas cercano) y aquí ya **interpola** (**bilinear**)

Patrón de fondo

Descripción generada automáticamenteImagen que contiene Patrón de fondo

Descripción generada automáticamente

Los mipmaps vienen con las tarjetas gráficas, vienen ya de por sí.

Imagen que contiene colgando, vidrio, tabla, pájaro

Descripción generada automáticamenteEn un sampling (con una muestra) podemos elegir si queremos bilinear, trilinear, ¿?

Aquí interpolamos los 4 píxeles

Los programadores podemos bajar a bajo nivel y decir exactamente el sampling que queremos o decirle aquí y quiero un 0, por ejemplo.  
Puedo meter un Bias: al calculo numero hecho podemos sumarle una constante

Gráfico

Descripción generada automáticamenteImagen que contiene colorido, edificio, brillando, naranja

Descripción generada automáticamente

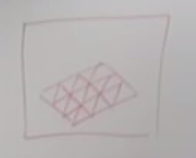
También podemos decirle que tome más muestras (activando el Ansiotropy). Cuesta más, pero es mejor.

Un dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza bajaComo leer de la Memoria era muy lento, se creó la VRAM (video RAM). Desde la CPU podemos escribir cosas en la VRAM pero no podemos leer (cuesta muchísimo tiempo).

Ahora vamos a pintar este plano:

Calendario

Descripción generada automáticamente

**Vertex Buffer**: zona de memoria con vértices. (tengo el plano)

**Index Buffer**: que veíamos el otro día (tengo los índices de los vertices)

**Vertex Shader**: se ejecuta para cada vértice. Hace que el vértice 0 del plano aparezca en la Coordenada X,Y de la pantalla.

**Pixel/Fragment Shader**: Una vez tengo los 3 vértices hay que pintar los pixeles del triángulo.

Imagen que contiene objeto, reloj

Descripción generada automáticamente

Entre ambos hay una rasterización

Pizarrón blanco con texto en letras negras sobre fondo blanco

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamentegametechdev.github.io/Intel-Texture-Works-Plugin/

Tabla

Descripción generada automáticamenteY nos copiamos el archivo aquí

Y en la carpeta Plugins otra cosa para los plugins

## DDS

* **Formato**Interfaz de usuario gráfica, Texto

  Descripción generada automáticamente
  + Precisión
  + Num canales
  + Num bits
  + Compresión

Ejemplos:

* + - BC1 RGB 4 bpp

Texto, Sitio web

Descripción generada automáticamente

* Filter Types

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Compresión

Y cuando acabamos, me guarda una textura

# LUT

Hay texturas 3D que se usan para las **LUT (Look up table)**

Que se usa para hacer

* Recolor tinte
* Color Grading

Texto

Descripción generada automáticamente

Se puede ver todo en esta pagina



# Cube Map Tortue

Representan las 6 caras del cubo.

Las UV es una dirección. Es muy cómodo tener un cube map. Y estas texturas también tienen sus Mipmaps.

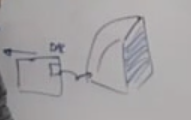
Dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza media

# FPS

A 60Hz

Tenemos 166ms para generar mi fotograma. Antes había un cañón de electrones.



Aquí había un problema pq el usuario podría ver en un momento dado un fotograma a mitad. Si yo cambio de fotogramas cuando el cañon de electrones va por la mitad. Para que no pase, tenemos que cambiar de fotogramas solo cuando el cañon de electrones ha llegado al final (Vertical Sync se llama)

Un hombre en frente de una computadora

Descripción generada automáticamente con confianza media

Entonces cuando llegamos al final, básicamente esperamos.

Imagen que contiene Texto

Descripción generada automáticamente

Ahora en el fotograma C tarda más ms en generarse.

Entonces se sigue viendo B, porq todavía no estaba listo

Un dibujo de un pizarrón blanco

Descripción generada automáticamente con confianza baja

En VR, hay que subir los HZ a 90 y encima hay que hacer los 2 ojos, y mas resolución.

**VFR**: ¿?

**Antialiasing**: (…¿?) y luego salieron filtros de antialiasing. Luego ha ido evolucionando a los samplers que hemos visto antes (temporal antialiasing). Suavizan la imagen.