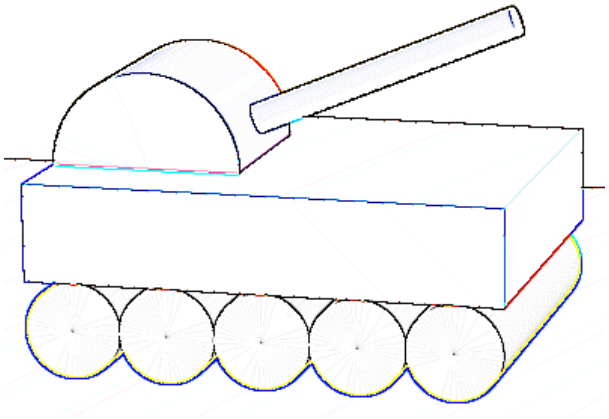


Tema 9: Procesamiento de Imagen

José Ribelles

Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I

SIU020 - Síntesis de Imagen y Animación

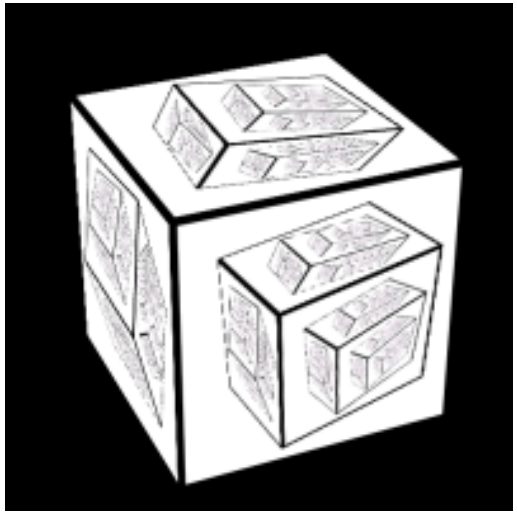
Contenido

Hoy veremos...

Introducción

A saco con los píxeles...

- Combina el dibujo a textura con la potencia del procesador de fragmentos.
- Cambia fácilmente el brillo, la saturación o el contraste.
- Aplica filtros para suavizar, perfilar o detectar bordes.
- Y como no, antialiasing!!
- Es un tema muy práctico, con muchos ejemplos.



Hoy veremos...

Antialiasing

¿Qué es el Aliasing?

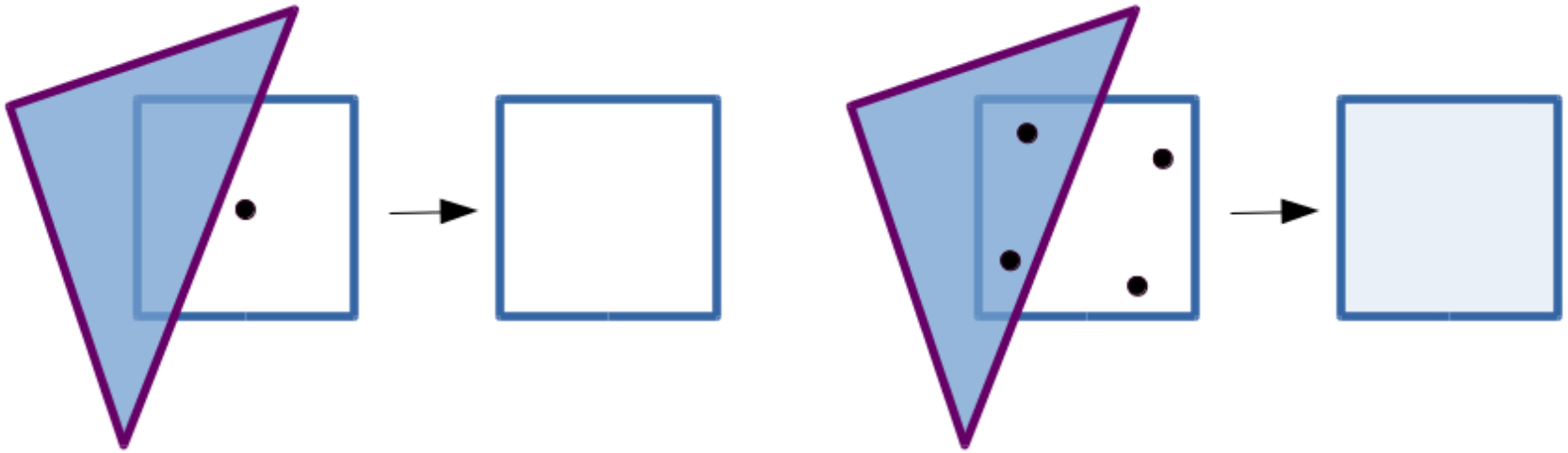
- Aparece siempre que discretizas una entidad continua.
- Principalmente en lineas y aristas de polígonos, pero también en las texturas.
- Fácilmente observable, quizá aún más en escenas animadas.
- La razón, un píxel pertenece o no a una entidad (punto, línea o polígono).

Supersampling

¿En qué consiste?

- Toma más muestras y mézclalas!
- Define un patrón de muestreo.
- Y suma las muestras con pesos para obtener el color del píxel p.

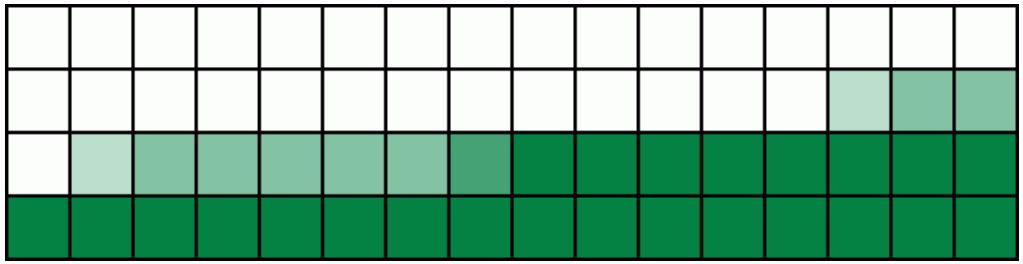
$$p(x, y) = \sum_{i=1}^{n} w_i c(i, x, y) \tag{1}$$



Supersampling

Full-Scene Anti-Aliasing o FSAA

- Es sin duda el más simple.
- Necesitas un framebuffer x veces mayor, siendo x el número de muestras.
- Y lo mismo con el buffer de profundidad.
- Procesa cada muestra de manera independiente, por lo que es muy costoso.



Multisampling

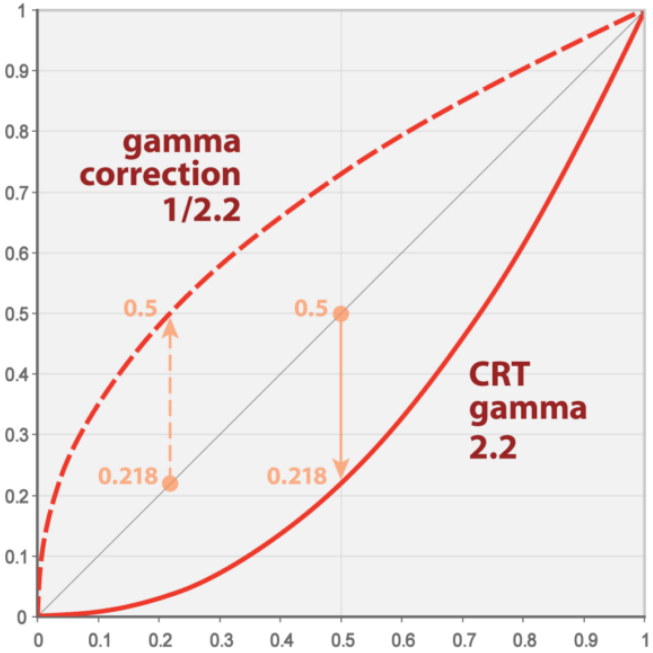
Multi-Sampling Anti-Aliasing o MSAA

- El Fragment Shader sólo se ejecuta una vez por píxel.
- Cada píxel se muestrea x veces para saber si cada muestra cae o no dentro de la entidad.
- En OpenGL es automático, el programador sólo lo ha de habilitar.
- En WebGL no hay, pero lo proporciona el navegador y por defecto está habilitado en el canvas.

Corrección Gamma

¿Qué es?

- Los monitores no tienen una respuesta lineal con los valores de intensidad de los píxeles.
- Esto produce que veamos las escenas un poco más oscuras o no tan brillantes como esperamos.



Corrección Gamma

¿Qué podemos hacer?

- La intensidad percibida es proporcional a la intensidad del píxel elevada a Gamma.

$$P = I^{\gamma} \tag{2}$$

- Normalmente Gamma está entre 2.0 y 2.4.
- Así, la correción Gamma consiste en contrarrestar este efecto:

$$P = (I^{\frac{1}{\gamma}})^{\gamma} \tag{3}$$

- En el Fragment Shader:
  $fragmentColor = vec4(pow(myColor, vec3(1.0/Gamma)), 1.0);$
- Donde *Gamma* debería ser Uniform ya que depende del monitor.

Ver ejemplo en el *AulaVirtual*

Ejemplo de Corrección Gamma

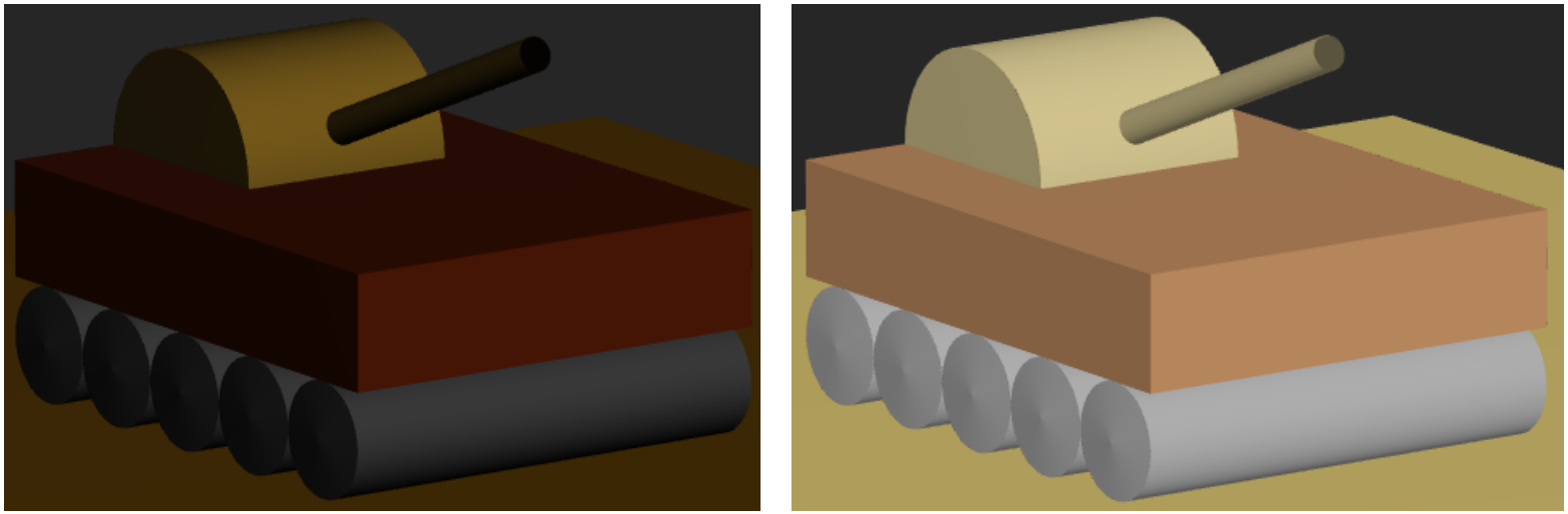


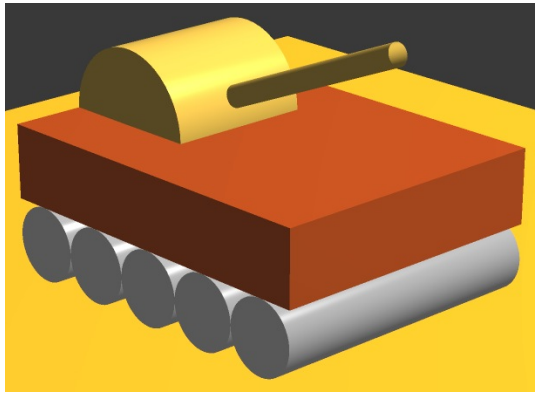
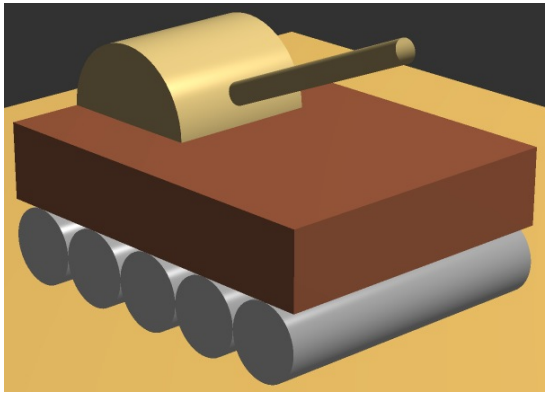
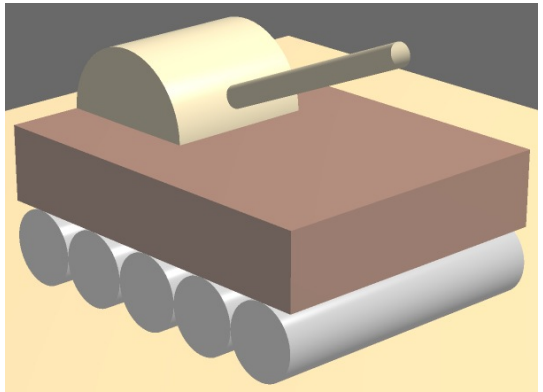
Figura: Izquierda, Gamma = 1.0; derecha, Gamma = 2.2

Hoy veremos...

Postproceso de Imágenes

¿Qué es?

- Operaciones que típicamente se realizan sobre la imagen resultado.
- Hoy en día se pueden implementar con facilidad en el Fragment Shader.
- Incluso se pueden realizar al mismo tiempo sin necesidad de realizarlo a posteriori.
- Pero si no es así, siempre puedes dibujar a textura y ...
- Lo hagas como lo hagas, en la GPU estas operaciones son muchísimo más rápidas que en la CPU.

Brillo, Contraste y Saturación

Brillo

Escala el valor de color de cada píxel.

```
fragmentColor = vec4(myColor * Brillo, 1.0);
```

Contraste

Mézclalo con un valor de gris.

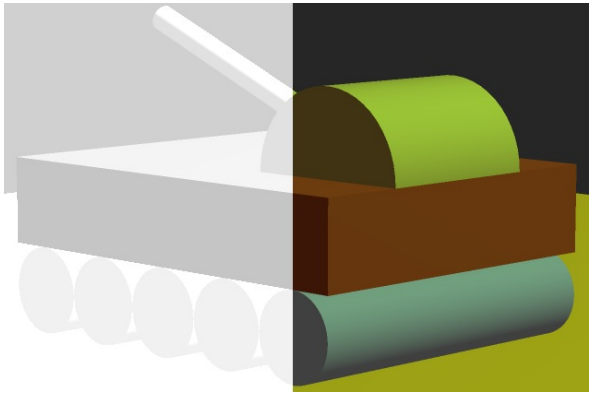
```
fragmentColor = vec4(mix(LumiMedia, myColor, Contraste), 1.0);
```

Saturación

Mézclalo con un valor de intensidad específico para cada píxel.

```
vec3 lumCoeff   = vec3(0.2125, 0.7154, 0.0721);
vec3 Intensidad = vec3(dot(myColor, lumCoeff));
fragmentColor   = vec4(mix(Intensidad,myColor,Saturacion), 1.0);
```

Brillo, Contraste y Saturación



¿Algo más?

Por supuesto, por ejemplo:

- Utiliza una textura compleja para indicar sobre qué zonas aplicar el efecto.
- O utiliza un patrón para indicar a qué píxeles aplicarlo.
- O alguna condición, como que estén en sombra, formen parte de un brillo o que simplemente estén en un rango de color.

Ver ejemplo en el *AulaVirtual*

Convolución

¿Qué es?

- Es una operación matemática fundamental en procesamiento de imágenes.
- Consiste en calcular para cada píxel la suma de productos entre la imagen fuente y una matriz mucho más pequeña a la que se le denomina filtro de convolución.
- Lo que la operación de convolución realice depende de los valores de dicho filtro.
- Necesitas obligatoriamente realizarlo sobre la imagen ya calculada.

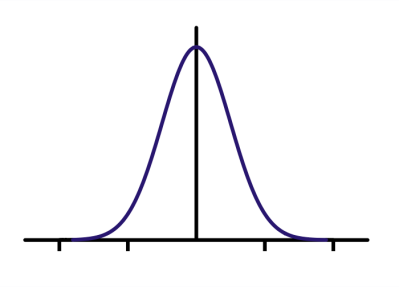
$$H(x,y) = \sum_{j=0}^{height-1} \sum_{i=0}^{width-1} F(x+i, y+j) \cdot G(i,j) \tag{4}$$

Convolución

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Suavizado

- Conocido por su término en inglés Blur o Smooth.
- Reduce el ruido de la imagen.
- Bueno para regiones de color sólido, pero emborrona también las aristas.
- Utiliza en su lugar un filtro de Gauss, más peso a los valores cerca del centro.

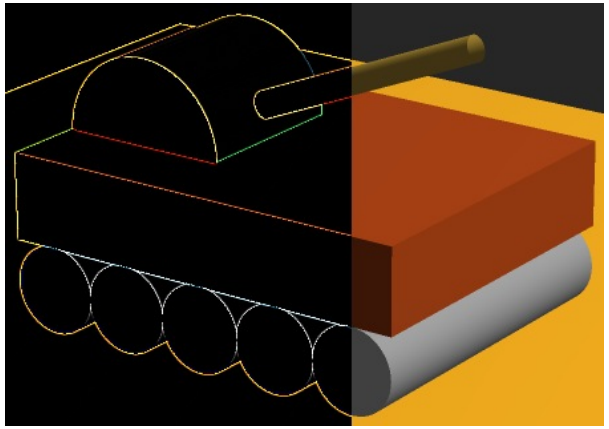


Convolución

-1	-1	-1
-1	8	-1
-1	-1	-1

Detección de bordes

- Detecta discontinuidades de intensidad.



Ver ejemplo en el *AulaVirtual*

Convolución

Perfilar

- Conocido en inglés por Sharpen.
- Utiliza primero un filtro para la detección de bordes.
- Suma entonces el resultado a la imagen original.
- Aplica antes un escalado al resultado del filtro.

```
fragmentColor =  colorSum * ScaleFactor +
                 texture2D(myTexture, texCoords);
```