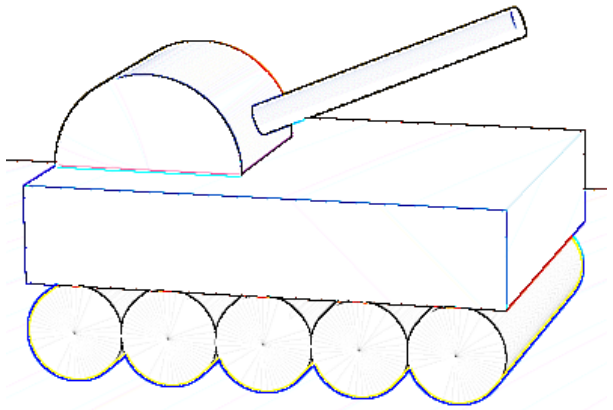


Tema 9: Procesamiento de Imagen

José Ribelles

Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I

VJ1221 - Informática Gráfica

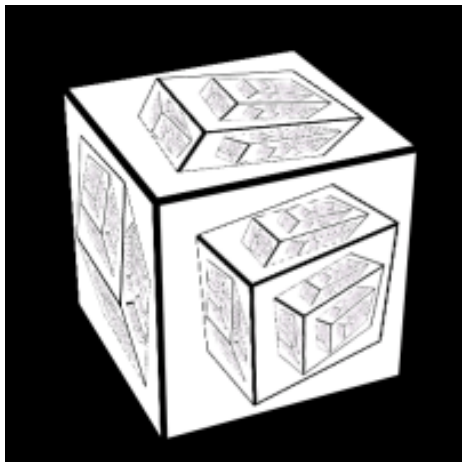
Contenido

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?

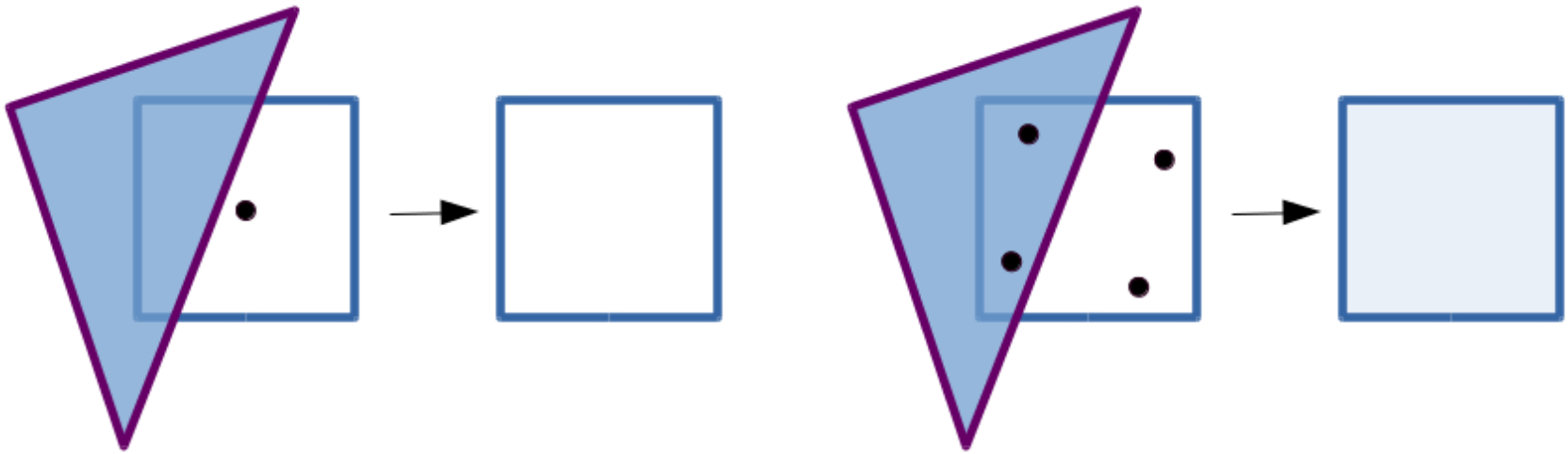
- Aparece siempre que discretizas una entidad continua.
- Principalmente en lineas y aristas de polígonos, pero también en las texturas.
- Fácilmente observable, quizá aún más en escenas animadas.
- La razón, un píxel pertenece o no a una entidad (punto, línea o polígono).

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.

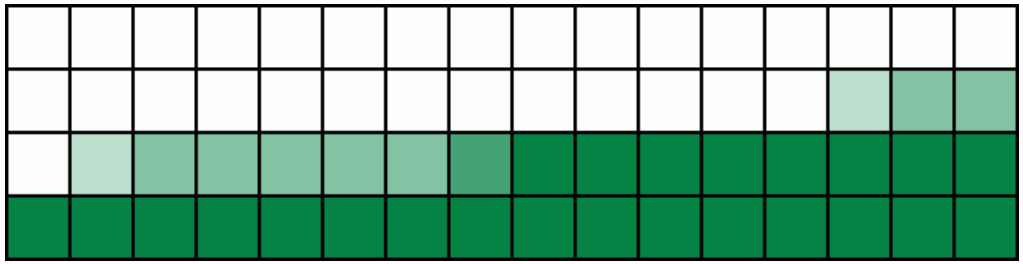
$$p(x,y) = \sum_{i=1}^{n} w_i c(i,x,y) \tag{1}$$



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

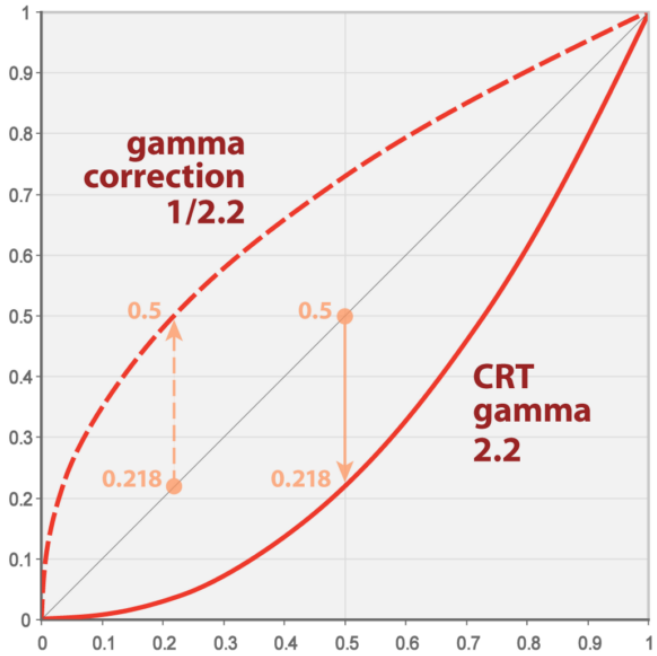
Multi-Sampling Anti-Aliasing o MSAA

- 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

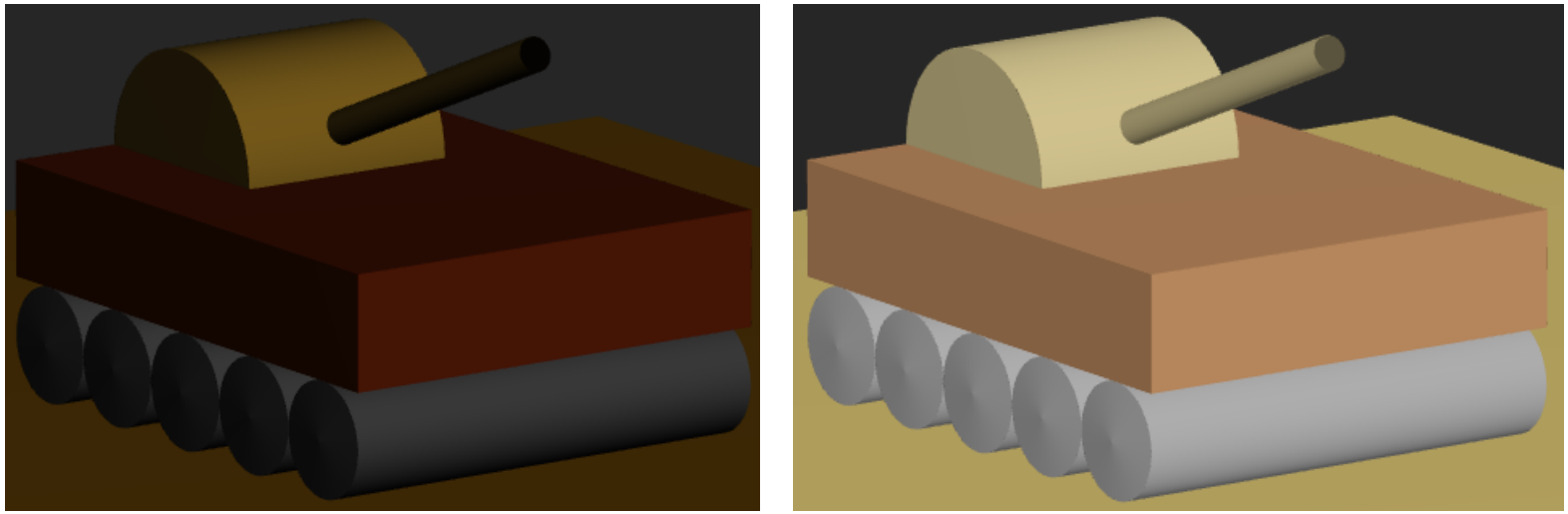


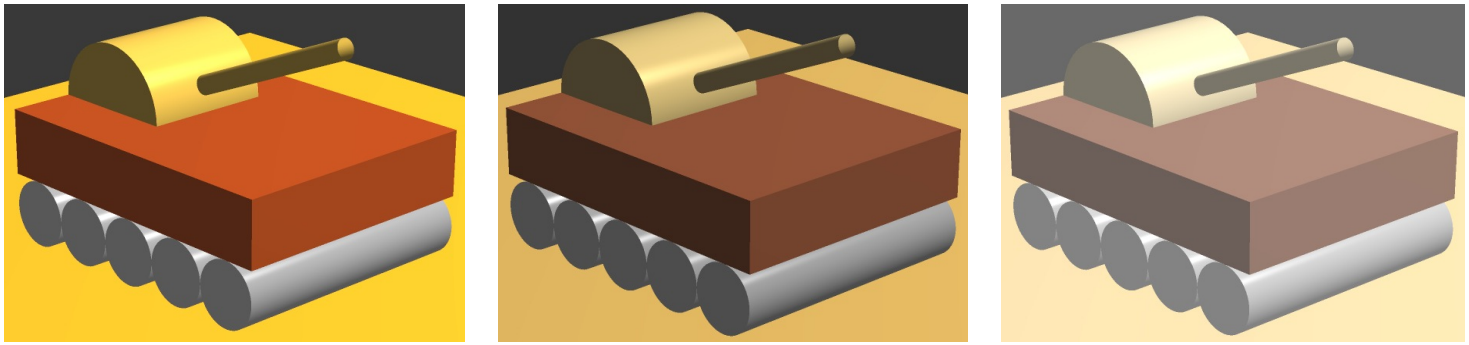
Figura: Izquierda, Gamma = 1.0; derecha, Gamma = 2.2

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.

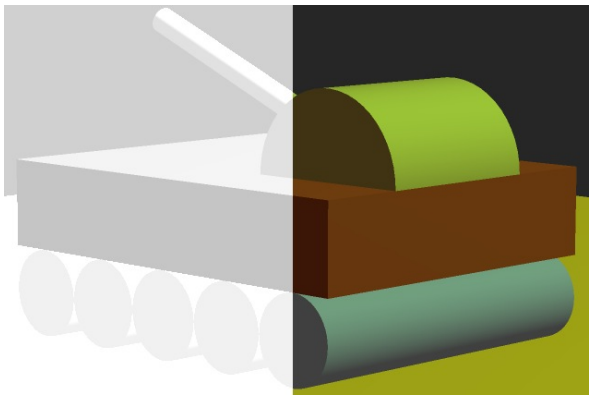
```
fragmentColor = vec4(mix(LumiMedia, myColor, Contraste), 1.0);
```

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.

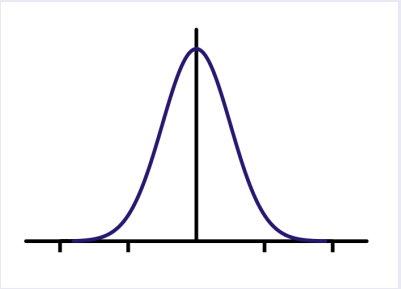
$$H(x,y) = \sum_{j=0}^{height-1} \sum_{i=0}^{width-1} F(x+i, y+j) \cdot G(i,j) \qquad (4)$$

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.

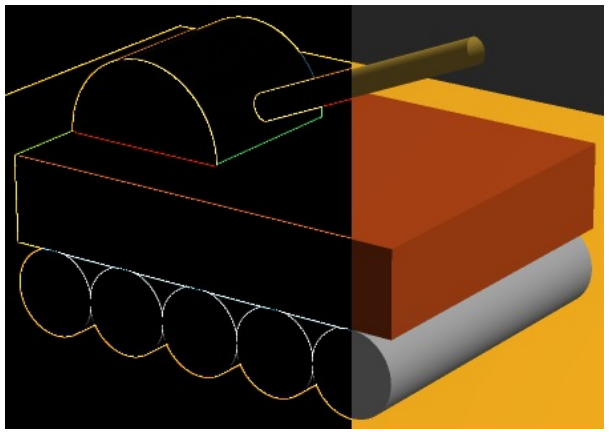


Convolución

-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);
```

Hoy veremos...

Tone mapping

High Dynamic Range Imaging

- Las escenas reales presentan un rango de luminancia mucho mayor que los 255 valores por componente que admite un dispositivo.
- HDR, en fotografía, consiste en capturar un mayor rango de intensidades del posible en una sola foto.

Tone mapping

Tone mapping

- Es el proceso por el que a partir de un amplio rango de valores se comprime en uno más pequeño apropiado al dispositivo.
- En informática gráfica, consiste en mapear un rango arbitrario de valores a un rango de 8 bits.
- Si por ejemplo tenemos una luz muy brillante en la escena:
  - si simplemente se eliminan las intensidades mayores que 1 tendremos una escena ... muy blanca.
  - si comprimimos linealmente las intensidades al rango 0..255, las zonas oscuras aparecerán muy muy oscuras.

Tone mapping

¿Cómo hacerlo?

1. Dibuja la escena contra una textura de alta resolución (32 bits por componente).
2. Calcula la luminancia logarítmica media.
3. Dibuja un rectángulo con la textura obtenida, y para cada píxel aplica el operador de Tone Mapping.

Paso 2

$$\bar{L}_w = exp(\frac{1}{N}\sum_{x,y} ln(0.0001 + L_w(x,y))) \quad (5)$$

Tone mapping

Paso 3

- Operador de Tone Mapping:

$$L_d(x,y) = \frac{L(x,y)(1 + \frac{L(x,y)}{L_{white}^2})}{1 + L(x,y)} \tag{6}$$

$$L(x,y) = \frac{L_w(x,y)}{\bar{L}_w(x,y)} \cdot a \tag{7}$$

- y $a \in [0.18..0.72]$ (como el valor de exposición de la cámara)
- y L_{white} un valor de luminancia máxima (configurable) para el caso en el que hayan píxeles excesivamente brillantes.
- Para operar con la luminancia es necesario hacer un cambio del espacio de color de *RGB* a *CIExyY*, en el que la Y es la luminancia.
- No es una ciencia exacta, hay que jugar con los parámetros hasta encontrar el aspecto deseado.

Bloom

¿Qué es?

- Efecto que produce que las partes brillantes se extiendan más allá de sus límites.

Bloom

¿Cómo conseguirlo?

1. Dibuja contra una textura.
2. Extrae las partes de la imagen que son más brillantes que un determinado umbral. Dibuja un rectángulo con la textura obtenida, y para cada píxel:

```
vec4 val = texture(myTexture, texCoord);
if ( luminance(val.rgb) > LumThreshold )
  fragmentColor = val;
else
  fragmentColor = vec4(0.0);
```

   En este punto genera una imagen de tamaño mucho menor (un octavo por ejemplo).
3. A esta nueva textura resultado aplica un filtro de emborronado.
4. Dibuja un rectángulo texturado con la suma de ambas texturas.