

Shaders

Gráficas por Computadora (CC2018) - 2026

Shaders

Semestre 02, 2026

Usos de los shaders

La GPU no trae efectos integrados. Cada efecto visual de una escena es un shader distinto o una variante de uno.

- Animar geometría en la GPU, como olas, telas y vegetación.
- Decidir dónde y con qué detalle se calcula la iluminación.
- Combinar varias texturas en un mismo material.
- Generar terreno, nubes, agua y fuego sin imágenes.
- Producir estilos no realistas, como el de una caricatura.
- Procesar la imagen terminada, como desenfoque o bordes.

Un programa por efecto

Un motor no usa un único shader. Cada material, cada tipo de luz y cada pase de render requiere su propio programa, y las combinaciones de opciones multiplican el número de variantes.

Un mismo material se compila en una variante con sombras y otra sin ellas, en una con normal map y otra sin él. Un motor comercial compila miles de variantes. Compilarlas durante la partida produce pausas visibles entre cuadros, conocidas como shader stutter, y por eso los motores las compilan por adelantado.

Shaders

Lenguajes de shaders

Lenguaje	API	Plataforma
GLSL	OpenGL, Vulkan	Multiplataforma
HLSL	DirectX	Windows y Xbox
MSL	Metal	macOS e iOS
WGSL	WebGPU	Navegador

Todos comparten el mismo modelo, con tipos vectoriales y matriciales nativos y un punto de entrada por etapa. El driver compila el programa en tiempo de ejecución para la GPU instalada, por eso un mismo shader corre en tarjetas de fabricantes distintos.

Etapas programables adicionales

Además del vertex y el fragment shader, el pipeline moderno admite tres tipos de shader más.

Shader	Se ejecuta una vez por	Produce
--------	------------------------	---------

Tessellation	Parche de la malla	Vértices nuevos que subdividen el parche
Geometry	Primitiva	Cero o más primitivas nuevas
Compute	Elemento de trabajo	Datos arbitrarios en buffers, fuera del pipeline gráfico

El tessellation y el geometry shader son etapas opcionales entre el vertex y el fragment shader. El compute shader no forma parte del pipeline gráfico, porque se usa para cálculo general como física de partículas o simulación de fluidos.

Programa de shaders

El programa de shaders es la unidad que resulta de compilar y enlazar los shaders de cada etapa.

- Las salidas del vertex shader deben coincidir en nombre y tipo con las entradas del fragment shader, o el enlace falla.
- Cambiar de material suele implicar cambiar de programa, lo que tiene un costo en la CPU.

Modelo de ejecución

Ejecución en paralelo

Cada invocación de un shader es independiente de las demás. La GPU puede ejecutarlas en cualquier orden y al mismo tiempo, porque ninguna depende del resultado de otra.

- Un fragmento no puede leer el color que calculó su vecino.

- Un fragmento solo escribe en su propia posición de pantalla.
- Una invocación no recuerda nada del cuadro anterior.

Lo que una invocación necesita saber del resto del mundo llega por uniforms y texturas, que son de solo lectura durante el draw call.

Warps

Un warp es un grupo de invocaciones que la GPU ejecuta en bloque con una sola unidad de control. Todas ejecutan la misma instrucción al mismo tiempo, cada una con sus propios datos.

NVIDIA los llama warps y los forma con 32 invocaciones. AMD los llama wavefronts y usa 32 o 64. Los fragmentos de un warp son vecinos en pantalla, porque así comparten datos de textura en caché.

Divergencia

La divergencia ocurre cuando las invocaciones de un mismo warp toman ramas distintas de una condición. El warp ejecuta una rama con las invocaciones de la otra desactivadas, y luego la otra.

$$t_{\text{warp}} = t_{\text{if}} + t_{\text{else}}$$

Una condición que vale igual para todo el warp, como una que depende solo de un uniform, no produce divergencia y no tiene costo extra. El costo aparece en los warps donde la condición cambia de valor, que en una imagen suelen ser los que cruzan el borde de una figura.

Funciones de forma

Las funciones de forma son funciones matemáticas que todo lenguaje de shaders trae integradas para transformar un valor en otro sin usar condiciones. Son la herramienta básica para evitar la divergencia.

$$\begin{aligned} \text{step}(e, x) &= \begin{cases} 0 & x < e \\ 1 & x \geq e \end{cases} \\ \text{mix}(a, b, t) &= a(1 - t) + b, t \end{aligned}$$

El resultado de step usado como peso de mix elige entre dos colores sin escribir un if. Todas las invocaciones ejecutan exactamente las mismas instrucciones, así que el warp nunca se divide.

Smoothstep

Smoothstep es una transición suave entre cero y uno a lo largo del intervalo de (e_0) a (e_1) , con derivada nula en los dos extremos.

$$\begin{aligned} t &= \text{clamp}\left(\frac{x - e_0}{e_1 - e_0}, 0, 1\right) \\ \text{smoothstep}(e_0, e_1, x) &= t^2(3 - 2t) \end{aligned}$$

Un borde hecho con step queda dentado, porque cada fragmento es completamente de un color o del otro. El mismo borde con smoothstep de uno o dos píxeles de ancho mezcla los colores en la frontera y se ve suavizado.

Modelos de sombreado

Un modelo de sombreado es la estrategia que decide en qué punto de la superficie se evalúa la ecuación de iluminación y cómo se reparte el resultado sobre el resto del triángulo.

La ecuación de iluminación es la misma en los tres modelos, que son flat, Gouraud y Phong shading. Conviene no confundir Phong shading, que es un modelo de sombreado, con el modelo de reflexión de Phong visto en Materiales,

que es la ecuación. Comparten el autor, no el concepto.

Flat shading

La iluminación se evalúa una vez por cara con la normal de la cara, y todos los fragmentos del triángulo reciben ese mismo color.

Es el modelo más barato, con una evaluación por triángulo. Las facetas quedan visibles, lo que sirve para un estilo low poly y no para superficies curvas.

Gouraud shading

La iluminación se evalúa en el vertex shader con la normal de cada vértice, y la rasterización interpola los tres colores resultantes sobre el triángulo.

Elimina las facetas con un costo de una evaluación por vértice. El problema está en el brillo especular. Si el reflejo cae en el interior de un triángulo, ningún vértice lo registra y se pierde. Al mover la luz, el brillo aparece y desaparece saltando de vértice en vértice.

Phong shading

La rasterización interpola las normales de los vértices, y el fragment shader evalúa la iluminación en cada fragmento con su normal interpolada.

El brillo especular queda redondo y se desplaza de forma continua al mover la luz o la cámara. La normal interpolada pierde su longitud unitaria en el interior del triángulo y debe normalizarse otra vez en el fragment shader.

Comparación de modelos

Modelo	Etapas de la luz	Lo que se interpola	Especular
--------	------------------	---------------------	-----------

Flat	Una vez por cara	Nada	Por cara
Gouraud	Vertex shader	El color	Se pierde o salta
Phong	Fragment shader	La normal	Correcto

El costo depende del tamaño de los triángulos. Un triángulo que cubre ocho fragmentos requiere tres evaluaciones con Gouraud y ocho con Phong. Si un triángulo cubre menos de un píxel, la relación se invierte y evaluar por vértice resulta más caro que evaluar por fragmento.

Iluminación por fragmento

Blinn-Phong

Blinn-Phong es la variante del modelo de Phong que cambia solo la componente especular. Mide el brillo con el vector medio (H) entre la luz y la cámara, en lugar del rayo reflejado (R).

$$I_s = k_s \max(0, N \cdot H)^\alpha \quad H = \frac{L + V}{\|L + V\|}$$

Calcular (H) cuesta menos que reflejar (L), y el brillo no se corta en ángulos rasantes. Es el modelo por defecto del pipeline fijo de OpenGL.

Texturas en el shader

Wrapping

El wrapping es la regla que decide qué texel devolver cuando la coordenada UV cae fuera del rango de cero a uno. Se configura por separado para cada eje.

Modo	Comportamiento
Repeat	Usa la parte fraccionaria y repite la imagen.
Mirrored repeat	Repite la imagen reflejada en cada repetición.
Clamp to edge	Estira el último texel del borde.
Clamp to border	Devuelve un color de borde fijo.

Repeat es el que permite cubrir un piso entero con una baldosa pequeña. Clamp to edge evita que una textura que no se repite arrastre en su borde los píxeles del lado opuesto.

Filtrado

El filtrado es la regla que calcula el color cuando la coordenada no cae justo en el centro de un texel. Importa sobre todo en la magnificación, cuando un texel cubre varios píxeles.

- Nearest: toma el texel más cercano. El resultado se ve pixelado.
- Linear: mezcla los cuatro texels vecinos según la distancia.

$$c = \frac{1}{4} \left(\frac{1}{s} \left(c_{00} + c_{10} + c_{01} + c_{11} \right) + s \right)$$

Nearest es la opción correcta para arte de píxeles, donde los bordes duros son intencionales. Linear es la opción habitual para fotografías y materiales.

Minificación y aliasing

Cuando la superficie se aleja, un solo píxel cubre muchos texels. Si el sampler toma una sola muestra, el color salta de un píxel al siguiente y aparecen patrones

de moiré que parpadean al moverse.

Es el mismo fenómeno que el aliasing de las líneas, aplicado a la textura. Promediar todos los texels de la huella en cada fragmento sería correcto, pero demasiado caro.

Mipmaps

Los mipmaps son una cadena de copias de la textura, cada una con la mitad del tamaño de la anterior y ya promediada. El sampler lee del nivel cuyo tamaño de texel se parece más a la huella del píxel.

$$\lambda = \log_2 \rho \approx 1 + \frac{1}{4} + \frac{1}{16} + \dots = \frac{4}{3}$$

(ρ) es la cantidad de texels que cubre un píxel a lo largo de un eje, y (λ) es el nivel que se lee. La cadena completa ocupa un tercio más de memoria que la textura original, y a cambio elimina el moiré y mejora el uso de la caché, porque los niveles pequeños caben enteros en ella.

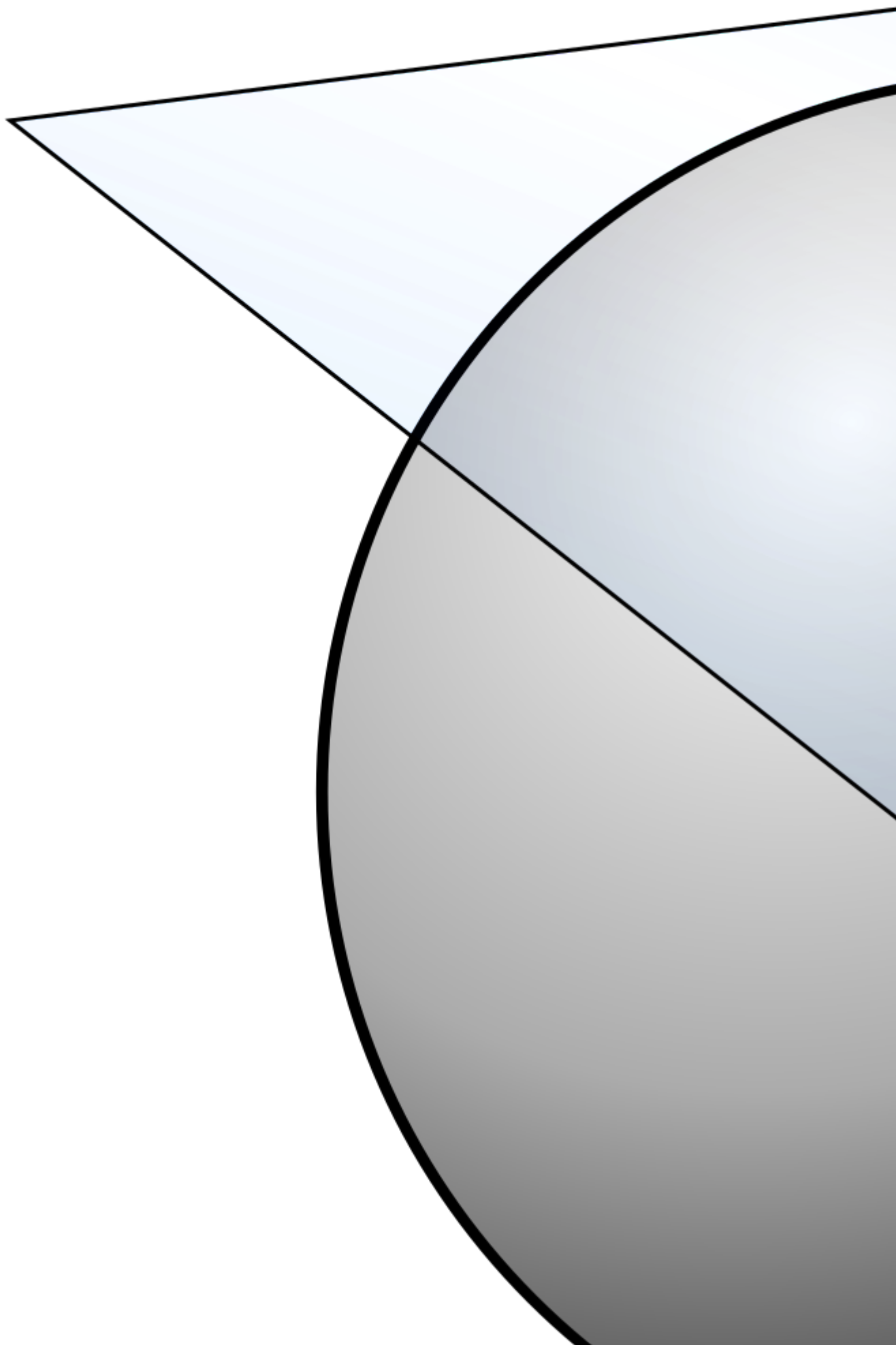
Texturas como datos

Una textura guarda cuatro números por texel, y el shader decide qué significan.

- Albedo: el color base del material.
- Normal map: una normal por texel, codificada en RGB.
- Height map: la altura de la superficie en cada punto.
- Roughness: qué tan concentrado es el brillo especular.
- Ambient occlusion: cuánta luz ambiente llega a cada rincón.
- Máscara: qué zonas reciben un efecto y cuáles no.

Normal mapping

El normal mapping reemplaza la normal interpolada por una leída de textura, para simular relieve sobre una superficie plana.



$$N = \text{normalize}(\text{big}(TBN, (2c - 1)\text{big}))$$

El color (c) va de cero a uno y se lleva al rango de menos uno a uno. La matriz (TBN), formada por la tangente, la bitangente y la normal del vértice, orienta esa normal según la superficie. La geometría no cambia, solo la respuesta a la luz, así que la silueta del objeto sigue siendo plana.

Shaders procedurales

Un shader procedural calcula el color a partir de una fórmula, en lugar de leerlo de una imagen. La entrada suele ser la coordenada UV o la posición del fragmento.

- Resolución ilimitada, porque se evalúa en cada fragmento.
- No ocupa memoria de textura.
- Se controla con uniforms y puede animarse con el tiempo.
- Cuesta instrucciones en cada fragmento, cada cuadro.

La UV como lienzo

Multiplicar la UV por (n) y quedarse con la parte fraccionaria divide la superficie en (n) celdas que repiten el rango de cero a uno. Cualquier figura dibujada en una celda se repite en todas.

$$f = \text{fract}(n, u) \quad \text{franja} = \text{step}(0.25, f) - \text{step}(0.75, f)$$

Un tablero combina una franja en (u) con otra en (v), y un círculo compara la distancia al centro de la celda con un radio. Al reemplazar step por smoothstep los bordes se suavizan.

Ruido

El ruido es una función pseudoaleatoria y continua, donde puntos cercanos devuelven valores parecidos. Es la base de nubes, terreno, mármol, fuego y agua.

Un número al azar por fragmento produce estática, porque vecinos cercanos no guardan relación. Además, el shader no tiene generador de aleatorios. Usa una función hash, que devuelve siempre el mismo valor para la misma entrada, y eso es lo que permite que la imagen sea estable de un cuadro al siguiente.

Value noise

El value noise asigna un valor aleatorio a cada esquina de una cuadrícula y lo interpola entre esquinas con smoothstep, para que no se noten los bordes de las celdas.

$$r(x) = \text{mix}(h(i), h(i + 1), \text{smoothstep}(0, 1, f)) \quad i = \lfloor x \rfloor, f = x - i$$

En dos dimensiones se interpolan las cuatro esquinas de la celda, primero en (u) y luego en (v). El ruido de Perlin asigna gradientes aleatorios en lugar de valores, y elimina el aspecto de bloques que deja el value noise.

Fractal Brownian Motion

El fBm es la suma de varias capas de ruido, llamadas octavas, cada una con más frecuencia y menos amplitud que la anterior. Agrega detalle fino sobre la forma general.

$$f_{bm}(p) = \sum_{k=0}^{N-1} g^k; r \big(\ell^k p \big)$$

- Lacunaridad (ℓ): multiplica la frecuencia en cada octava, normalmente por 2.

- Ganancia (g): multiplica la amplitud en cada octava, normalmente por 0.5.

Cada octava cuesta una evaluación de ruido más, así que el número de octavas es un compromiso entre detalle y rendimiento.

Ejemplo de planeta procedural

Se requiere sombrear una esfera para que parezca un planeta con océanos, costas, vegetación y montañas nevadas, sin usar ninguna textura.

1. Altura. Evaluar un fBm sobre la posición del fragmento en la esfera.
2. Rampa de color. Comparar la altura con el nivel del mar, que es un uniform, y elegir agua, arena, pasto, roca o nieve con smoothstep.
3. Iluminación. Aplicar Lambert al terreno, brillo especular solo al agua y un halo de atmósfera en el borde, donde la normal casi no mira a la cámara.

Usar la posición en la esfera y no la UV evita la costura y el estiramiento en los polos que produce envolver una imagen plana alrededor de una esfera.

Efectos en el vertex shader

Desplazamiento de vértices

El desplazamiento de vértices mueve cada vértice antes de aplicar las matrices, en función de su posición, del tiempo o de una textura de altura.

- Olas sobre una superficie de agua.
- Banderas y telas movidas por el viento.
- Pasto y hojas que se mecen.
- Terreno levantado con un height map.

Ondas en la malla

Cada vértice se desplaza en la dirección de su normal según una onda que avanza con el tiempo, que llega como uniform.

$$p' = p + n \cdot A \sin(\omega, x + t)$$

(A) es la amplitud y (ω) la frecuencia de la onda. Al deformar la superficie, la normal original deja de ser válida. Hay que recalcularla con la derivada de la onda, o la iluminación no acompaña al movimiento.

El límite de resolución

El vertex shader no agrega detalle. Una onda más fina que la separación entre vértices no se puede representar, sin importar la fórmula. Para una onda de (c) ciclos la malla necesita al menos $(2c + 1)$ vértices en esa dirección.

- Subdividir la malla en la CPU, a costa de más vértices en memoria.
- Usar un tessellation shader, que genera vértices nuevos en la GPU solo donde hacen falta.
- Simular el detalle fino en el fragment shader con un normal map, sin mover la geometría.

Efectos en el fragment shader

Toon shading

El toon shading cuantiza la intensidad de la luz en pocas bandas planas, imitando el sombreado de una ilustración o de una caricatura.

$$I_{\text{toon}} = \frac{\lfloor I \cdot n \rfloor}{n}$$

(n) es el número de bandas. El brillo especular se trata igual, con un solo umbral que lo vuelve una mancha de bordes duros.

Rim light y contorno

El rim light es una luz que aparece en el borde de la silueta, donde la normal es casi perpendicular a la dirección de la cámara.

$$\text{rim} = \max(0, N \cdot V)^p$$

El mismo producto ($N \cdot V$) sirve para un contorno, pintando de oscuro los fragmentos donde cae por debajo de un umbral. El exponente (p) controla qué tan delgada es la franja.

Descarte de fragmentos

La instrucción discard del fragment shader abandona el fragmento sin escribir color ni profundidad. El píxel conserva lo que ya tenía.

- Recortar hojas, rejas y cabello a partir del canal alfa de una textura.
- Disolver un objeto comparando un ruido con un umbral que crece. Los fragmentos cuyo ruido está apenas por encima del umbral se pintan de un color brillante y forman el borde que se quema.

Un shader con discard impide que la GPU haga el depth test antes de ejecutarlo, porque no sabe de antemano si el fragmento sobrevivirá. Por eso se usa solo donde hace falta.

Post-processing

El post-processing agrupa los efectos que se aplican sobre la imagen ya renderizada y no sobre la geometría. La escena se dibuja en una textura y un

segundo pase la procesa.

1. Primer pase. Renderizar la escena en un framebuffer cuyo destino es una textura.
2. Segundo pase. Dibujar un rectángulo que cubre la pantalla, con un fragment shader que lee esa textura y escribe el resultado.

El costo del segundo pase depende solo de la resolución de la pantalla y no de la complejidad de la escena.

Filtros por píxel

Cada fragmento solo necesita su propio color.

$$Y = 0.2126R + 0.7152G + 0.0722B$$

- Escala de grises: la luminancia (Y) en los tres canales.
- Inversión: uno menos el color, canal por canal.
- Viñeta: oscurecer según la distancia al centro.
- Corrección de color: curvas por canal con una tabla.

Filtros por vecindad

El fragmento lee varios texels alrededor de su posición y los combina con un kernel de pesos.

$$K_{\text{blur}} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

El desenfoque promedia la vecindad. Sobel mide el cambio de luminancia en cada eje, y la magnitud de ese cambio resalta los bordes de la imagen. Este tipo de

filtro solo es posible en un segundo pase, porque en el primero un fragmento no puede leer a sus vecinos.

Dónde vive cada efecto

Efecto	Etapas	Se ejecuta una vez por
Ondas y desplazamiento	Vertex shader	Vértice
Gouraud shading	Vertex shader	Vértice
Phong shading y normal mapping	Fragment shader	Fragmento
Patrones, ruido, toon y disolución	Fragment shader	Fragmento
Filtrado y mipmaps	Unidad de textura	Lectura del sampler
Desenfoque y bordes	Segundo pase	Píxel de la imagen

Glosario

Término	Definición
Programa de shaders	Shaders de cada etapa compilados y enlazados como una unidad.

Término	Definición
Warp	Grupo de invocaciones que ejecuta la misma instrucción a la vez.
Divergencia	Ramas distintas dentro de un warp, que se ejecutan una tras otra.
Smoothstep	Transición suave entre cero y uno dentro de un intervalo.
Phong shading	Iluminación evaluada por fragmento con la normal interpolada.
Discard	Abandono de un fragmento sin escribir color ni profundidad.
Sampler	Uniform que da acceso a una textura con sus reglas de lectura.
Wrapping	Regla para coordenadas UV fuera del rango de cero a uno.
Mipmap	Cadena de copias reducidas y promediadas de una textura.
Normal map	Textura que guarda una normal por texel.
Ruido	Función pseudoaleatoria continua.
fBm	Suma de octavas de ruido con frecuencia creciente.