Me gustaría arrancar esta presentación enfocándome en un problema en particular.

Problema

Supongamos que tenemos una aplicación que es una red social. En esta pueden compartirse imágenes y videos. Y queremos hacerla más inclusiva y amigable para las personas daltónicas. Supongamos que podemos modificar la interfaz, ¿pero y los videos e imágenes?

También las podemos modificar, y lo haremos en tiempo real. Para ello recurriremos a los *Shaders*.

DIAPO

Qué son los shaders?

Simplificando, los pixel shaders (los cuales mostraremos hoy) son programas en los cuales sus instrucciones son ejecutadas de forma paralela por cada pixel de la pantalla. Eso significa que el código que escribis puede comportarse de manera diferente dependiendo de su posición en la pantalla. El hecho de que los shaders se ejecuten de forma paralela por cada pixel implica una ventaja muy grande, ya que al ser cada pixel una tarea, nos permite renderizar cosas en pantalla a una velocidad muy alta.

DIAPO

Estos tipos de shaders se encuentra al final o cerca del final en el pipeline de renderizado gráfico. Podemos imaginarlos como una función que recibe una posición en pantalla y devuelve un color.

Por lo tanto podemos reducir la salida de un shader a cuatro valores: r, g, b y a, donde representan los valores de rojo, verde, azul y transparencia.

DIAPO

El término “shader” fue introducido al público por Pixar en 1988 con el software RenderMan. Inicialmente los shaders eran usados para offline rendering, por ejemplo en Toy Story.

Mientras las GPUs evolucionaron, las principales bibliotecas gráficas tales como OpenGL o Direct3D comenzaron a darle soporte a shaders, permitiendo así la renderización de estos en tiempo real.

Son llamados shaders (traducido sería sombreadores) porque eran usados usualmente para controlar la iluminación y los efectos de sombreados de una escena, pero no hay ninguna razón por la que no puedan manejar otros efectos especiales.

Aquí es donde procesar en paralelo se vuelve una buena solución. En vez de tener un par de procesadores grandes y poderosos, o tubos, es mucho más inteligente tener muchos pequeños procesadores funcionando en paralelo al mismo tiempo. Eso es la GPU (Graphic Processor Unit).

Espacio de colores

Antes de pasar a temas más técnicos me detengo un poco para explicar qué es esto último que mencione.

Hay muchas maneras de representar colores, la más conocida y la usada por shaders es RGB, es decir una combinación de 3 valores, rojo, verde y azul, para identificar un color.

Existen otros como HSL, RYB, CMYK. Cada espacio de colores tiene ciertas ventajas que lo hacen conveniente de usar en diferentes aplicaciones. En nuestro problema a tratar traduciremos colores del espacio RGB a uno llamado LMS, el cual es un espacio de color que representa la respuesta de los tres tipos de conos del ojo humano (llamados así por sus picos de respuesta a longitudes de onda largas, medias y cortas). Este espacio de colores es apropiado para trabajar con conceptos como el daltonismo, razón por lo cual lo usaremos. representado por los tres tipos de conos del ojo humano.

DIAPO

Programando Shaders

Para programar shaders hoy utilizaremos shadertoy, así que aclaramos algunas cosas del lenguaje utilizado por este y la mayoría de los lenguajes de shaders antes de encarar el problema presentado:

- El lenguaje utilizado por los shaders es similar a C, tiene variables reservadas, funciones y tipos. Existe por ejemplo el tipo vec4, que es un tipo de variable de 4 dimensiones de punto flotante.

- Los shaders tienen una función main principal que recibe como entrada una posición de pixeles fragCoord y devuelven un color al final, el cual es guardado en la variable global reservada fragColor.

- Y como mencionamos, el espacio de colores utilizado para guardar la información del pixel es RGBA, en donde los valores se encuentran normalizados, eso significa que van desde 0.0 a 1.0

El valor que devolvemos, un color, es un vector de 4 componentes. Para acceder a estas componentes lo podemos hacer de las siguientes maneras:

```
vec4 vector;
```

```
vector[0] = vector.r = vector.x = vector.s;
vector[1] = vector.g = vector.y = vector.t;
vector[2] = vector.b = vector.z = vector.p;
vector[3] = vector.a = vector.w = vector.q;
```

Como pueden ver, podemos acceder a estos valores como si fuera un array, o como si fuera una estructura como en programación orientada a objetos.

Lenguajes de shaders como el usado por shadertoy nos permite además jugar un poco con esto, permitiéndonos usar por ej vector.gb para obtener un vector de dos componentes donde vector.gb=[vector.g,vector.b].

Cuando usamos vectores y funciones trigonométricas, vec2, vec3 y vec4 son tratados como vectores, incluso cuando representan colores.

DIAPO

Funciones de hardware

Dentro de los lenguajes de shaders algunas funciones matemáticas especiales son aceleradas vía hardware, es decir la matemática más compleja es solucionada directamente en el hardware en vez de resolverlo en el software. Eso significa que tendremos una velocidad extra en cálculos trigonométricos u operaciones de matrices que irán tan rápido como la electricidad. En pantalla pueden ver algunas de estas funciones.

(hablar primero de las constantes) DEMO HACIENDO GRADIENTES

En particular en shadertoy tenemos algunas constantes que nos permiten hacer los shaders más dinámicos. Si bien la forma de acceder a estas varia en el lenguaje de shaders que utilicemos, en shadertoy podemos usarlas directamente de la siguiente manera:

Necesitamos poder enviarle valores de entrada desde la CPU a todos los threads. Debido a la arquitectura de la GPU todos esos valores van a ser iguales (uniform) para todos los threads y de sólo lectura. En otras palabras, cada thread recibe la misma información y puede leerla pero no modificarla.

Variables como fragCoord que varían dependiendo el thread se les llama varying.

DEMO JUGANDO CON EL MOUSE O ALGO

Resolver el problema

Primero, tomamos el pixel del video a modificar y lo convertimos del espacio de colores RGB al LMS, utilizando una matriz de conversión correspondiente..

Pasaremos luego a simular el daltonismo elegido reduciendo los colores acordemente.

Luego, convertimos los colores afectados por el daltonismo de LMS a RGB usando la inversa de la matriz que utilizamos antes.

Por último, compensaremos el daltonismo desplazando las longitudes de onda de la parte del espectro invisible para el dicromático, hacia la parte visible.

DEMO

DIAPO

Debugging

Como los shaders son inherentemente paralelos, debugear no es una tarea sencilla. A menos que el framework lo permita (por ej si estamos programando shaders dentro de un engine), no tenemos una consola en la cual observar mensajes de error. Por lo tanto, no nos queda otra que utilizar la salida de los shaders para poder entender qué está pasando dentro del código. Así que recordemos, ¿que obtenemos al ejecutar shaders? Colores.

DIAPO

Probablemente hayan visto en algún juego o software de renderizado modos de debugging. Estos pueden expresar gran cantidad de información cuando son usados de forma correcta.

DIAPO

Por ejemplo, en la imagen en pantalla podemos ver un modo de debugging que nos muestra el normalmap de una escena. Un normalmap es un mapa de ‘‘ángulos’’ si se quiere, en donde se representa el ángulo de cada vértice en pantalla en relación a la cámara, con un color.

Luego, en la segunda imagen podemos ver un modo de debugging para observar únicamente la oclusión ambiental (las sombras) calculada.

DIAPO

Inconveniente

Un ejemplo de shaders en acción que están usando ahora es google meet. Aunque ahora nadie este usando la funcionalidad, google meet nos permite aplicar un efecto de desenfoque al fondo, o aplicar

una imagen plana detrás nuestro. Para hacerlo generan una máscara en tiempo real con una inteligencia artificial para aplicar estos efectos al fondo del sujeto.

Supongamos que queremos aplicar una imagen de fondo detrás nuestro en la cámara y aplicarle un desenfoque a esta imagen. Eso implicaría renderizar el fondo detrás nuestro y tomar pixeles cerca del cual estamos parados para realizar un promedio (o el algoritmo de desenfoque que queramos aplicar), pero al estar cada pixel renderizado de forma paralela, no podamos obtener su valor hasta después de renderizar el fondo, por lo cual necesitaremos esperar a que este se renderice primero para después poder aplicar el efecto de desenfoque.

A este concepto de esperar a que se termine de ejecutar una "parte" del shader para luego aplicar otra se los llama passes (pasadas). Es una forma de aplicar concurrencia a nuestros shaders. Necesaria para cuando necesitamos trabajar sobre información calculada en otros pixeles a parte del cual estamos parados. Para aplicar esta técnica basta con guardar la información procesada en un buffer para después leerla en otra pasada, como si fuera la primer pasada que estábamos realizando en el shader para daltónicos.

Que otras cosas se pueden hacer con shaders?

- Crear formas procedurales, patrones, texturas y animaciones.
- Procesar imagenes (desenfocado, filtros de color, lookup tables, etc.). Mencionamos como podemos aplicar la corrección de colores para daltónicos. Este tipo de correcciones pueden usarse en reproductores de video que soportan la ejecución de shaders, como MPC-HC.
- Simulaciones (El juego de la vida de Conway, ondas de agua, rebotes de rayos de luz, etc).

DIAPO

Como correr shaders:

Para programar y utilizar shaders se debe usar un lenguaje que pueda ser compilado y ejecutado en donde lo queremos usar, pero si quieren jugar un poco pueden probar shadertoy como estuve mostrandoles o alguno de los softwares en pantalla.

DIAPO

Extra para investigar que no me dio tiempo de hablar

Les menciono un par de cositas para seguir investigando a quién le interesa.

- Si quieren indagar un poco con ejercicios para meter mano y aprender sobre temas más complicados pueden buscar the book of shaders en internet, que aunque no me gusta mucho a mucha gente parece resultar la mejor referencia para aprender.

- Si guardamos el resultado obtenido por un shader en una textura, podemos ejecutar otro shader sobre el resultado del anterior, permitiéndonos así poder observar resultados de computaciones sobre otros píxeles diferentes al que estamos parados.
- El mezclados de colores es un tema muy grande con un montón de efectos que suele aplicarse en una gran variedad de temas, por lo que vale la pena verlo.