**Cosas a tener en cuenta para el tp**

**nota:**

4 tp ok

3 entrega en termino

-2 1 dia de demora

-1 2 dias de demora

0 mas

3 lo que le guste, codigo limpio,funciones extra, etc

**3 ENTREGAR EN TERMINO PARA TENER MAS NOTA con informe y tutti**

**Desarrollo del trabajo (buen código, informe bueno, pruebas sobre algunas imagenes copadas)**

**Plantear mejoras**

**Aportar todo lo mejor que se nos ocurra**

**Agregar cosas**

**-----------------------------**

**Procesamiento de Imágenes.**

**Recibimos una imagen como entrada, y la exportamos con otra.**

**Manejo de archivos - C**

**Assembler - procesamiento de imagenes.**

**GraphicsMagick para procesar las imagenes en rgb (vectores de bytes para las imagenes, 3 bytes por pixel).**

**gm convert -resize colxfil! (al no poner el signo de admiracion se fuerza deformando la imagen)**

**1er etapa: linea de comandos para interpolar las imagenes. (cuanto quiero de cada la 1er y 2da imagen es decir ) entre 0 y 1 (float) parametro p. Interpolacion lineal.**

**Para cada pixel o color calculamos la interpolacion**

**r = p × v 1 + (1 − p ) × v2**

**usar xmm para el procesamiento de imagenes.**

**Convertir nros a punto flotante.**

**Programar 2 funciones en C**

**int​ ​leer\_rgb(char​ ​\*archivo,​ ​unsigned​ ​char​ ​\*buffer,​ ​int​ ​filas,​ ​int​ ​columnas);**

**int​ ​escribir\_rgb(char​ ​\*archivo,​ ​unsigned​ ​char​ ​\*buffer,​ ​int​ ​filas,​ ​int​ ​columnas);**

**Usar malloc, revisar si malloc falla, etc**

**Pueden ser void estas 2 funciones**

**Programar en Assembler**

**void​ ​interpolar(int longitud, unsigned​ ​char​ ​\*img1,​ ​unsigned​ ​char​ ​\*img2,​ ​unsigned​ ​char​ ​\*resultado, float​ ​p);**

**Recorre las 2 imagenes**

**Compara cada pixel de cada color y los interpola de acuerdo al parametro p ( entre 0 y 1) y mete el resultado en la variable resultado.**

**Para procesar usaremos xmm (128 bits) -> 16 bytes. Entran 4 numeros de punto flotante de precision simple.**

**3) Escribir un programa en c usando los argv o sea los parametros.**

**Podemos también tirar errores por si no tira los parametros indicados, o sino mandar por la salida estandar si no nos indica el ultimo parametro para guardarlo en un archivo.**

**usar atof para converit p a float**

**Incluir en el informe imagenes de prueba y resultado de combinarlas.**

**Hacer los .bat para compilar y ejecutar todas las pruebas.**

**Con XMM podremos operar 4 numeros de punto flotante.**

**Como hacemos para meter los otros bytes consecutivos.**

**Los valores los procesamos de a 4. Con 4 bytes (double word) usamos**

**MOVD xmm0, memory (copia en la parte baja de todo el registro)**

**Lo que hay que hacer para convertir bien el nro entero a float para meter el float en el lugar correcto del registro hay que usar SHUFFLE**

**PSHUFB XMM0, mascara o indices a usar ( 0 a 15)**

**PSHUFB XMM0, xmm7 (vector que indica como van los numeros)**

**cvtdq2ps dest, origen (convert double quad word**

**operaciones de punto flotante**

**addps**

**subps**

**divps**

**mulps**

**Hacer shuffle del valor p, para multiplicad de una con el vector**

**para almacenar la imagen hay que convertir a bytes usando**

**cvtps2dq**

**Deja los bytes en las posiciones que separamos antes.**

**LO que hay que hacer es dejarlos juntos como antes.**

**Para eso hay que hacer movd de nuevo.**

**la dificultad esta en combinar las instrucciones cvt…. pshufb, etc.**

**EXTRA: HACER EN C TAMBIÉN Y MOSTRAR COMPARACION**

**Etapa 2.**

**CAMBIAR UN FONDO PLANO POR OTRO PARA INTERPOLAR 2 IMAGENES.**

**ES DECIR TENER LA POSIBILIDAD DE ELEGIR COLOR TRANSPARENTE.**

**Para eso vamos a implementar mas funciones en C.**

**void​ ​separar\_rgb(unsigned​ ​char​ ​\*rgb,​ ​int​ ​cantidad,​ ​unsigned​ ​char​ ​\*r,​ ​unsigned char​ ​\*g,​ ​unsigned​ ​char​ ​\*b);**

**void​ ​combinar\_rgb(unsigned​ ​char​ ​\*r,​ ​unsigned​ ​char​ ​\*g,​ ​unsigned​ ​char​ ​\*b,​ ​int cantidad,​ ​unsigned​ ​char​ ​\*rgb);**

**cantidad de pixel**

**Separar cada color en un vector.**

**vector r, vector g, vector b**

**Combinar todos en un vector solo {r,g,b,r,g,b,r,g,b}**

**Hacer en assembler**

**superponer(unsigned​ ​char​ ​\*figura,​ ​unsigned​ ​char​ ​\*fondo,​ ​int​ ​r,​ ​int​ ​g,​ ​int​ ​b, float​ ​t,​ ​unsigned​ ​char​ ​\*resultado);**

**Pasar color rojo verde y azul del fondo transparente, y aceptar colores parecidos que puedan ser transparentes, es decir una distancia determinada de ese color lo podemos aceptar como transparente también.**

**Para eso hacemos una resta o modulo entre 2 vectores**

**r1 g1 b1 color de la imagen, r g y b color transparente.**

**Si la distancia es menor o igual a t, el pixel es transparente.**

**Escribir el programa en c con los parametros necesarios.**

**PARCIAL 17/11.**

**REC 1/12**

**ASM SOLO EN 32 BITS**

**IMPRIMIR CODIGO FUENTE**

**.BAT PRUEBAS**

**ASSEMBLER**

**ACTUAL:**

**en xmm7 esta el patron PF A INT.**

**en xmm6 INT A PF**

**xmm5 PATRON CARGA P**

**xmm3 P**

**xmm4 1-P**

**xmm0 TIENE IMAGEN 1**

**xmm1 tiene imagen 2**

**xmm2 Nada**

**DESEADO**

**xmm0 TIENE IMAGEN 1**

**xmm1 tiene imagen 2**

**xmm2 P (transparencia)**

**xmm3 1-p**

**xmm4 patron carga p**

**xmm5 int a pf**

**xmm6 pf a int**

MODIFICACION DE LA PARTE 2

igual que antes, estan las funciones de c -> void separar\_rgb

y void combinar\_rgb

cambio la parte de assembler

///rt gt y bt es el color transparente

void gen\_mascara(unsigned char \*r, unsigned char \*g, unsigned char \*b, int cantidad, int rt, int gt, int bt, float t, unsigned char \*mascara)

void aplica\_mascara(unsigned char \*mascara, unsigned char \*canal1, unsigned char \*canal2, int cantidad, unsigned char \*resultado)

Como operar las mascaras

Tenemos que hacer operaciones logicas

Supongamos que tenemos 11100111 (mascara binaria en bytes)

aplicando la mascara con operaciones AND OR Y XOR

AND devuelve 1 cuando los 2 son 1, y si es 0 devuelve 0 -> devuelve X pero es seguro que las partes donde tenga 0 el byte devuelve 0 indistintamente.

FFFF00FF00 Mascara

xxxxxxxxxx Vector

XXXX00XX00 Resultado

Podemos usar PANDN

PANDN A B

Le asigna al parametro A el complemento del parametro B

De esta forma nos quedamos con la figura, y descartamos al fondo.

Podrìamos usarlo como último parametro.

Luego para combinar los 2 valores hay que hacer un POR

como hace gen\_mascara para generar la mascara

Las instrucciones de operacion de SIMD operan con numeros solamente, no tocan las FLAGS.

Dejan todo en 1 cuando la comparación es verdadero y cuando no todo en 0

CMPPS trabaja sobre double quadword con 4 doubles adentro 4 paquetes de 32 bytes (4 bytes)

CMPPS DEST, ORIGEN, BYTE PARA INDICAR TIPO DE OPERACION

BYTES: 0 -> CMP por igual

1 -> Menor

2 -> menor igual

4: Distinto

5: >=

3: No son comparables (Not A Number)

6: >

7: Estan ordenados.

0: OP  EQ;

1: OP  LT;

2: OP  LE;

3: OP  UNORD;

4: OP  NE;

5: OP  NLT;

6: OP  NLE;

7: OP  ORD;

Calcular distancia con la raiz cuadrada para sacar distancia euclediana.

Si la distancia es menor o igual a T entonces es transparente.

Mover partes de los 4 paquetes de 32 bits a otro paquete para la mascara.