Imágenes Bit Map Portable

HISTORIAL DE REVISIONES			
NÚMERO	FECHA	MODIFICACIONES	NOMBRE
v1.0.0	2017 Noviembre 21		CA

Índice

1. Introducción

importante
Práctica introductoria al examen final por lo que es necesario realizarla de forma **individual** para obtener el mayor rendimiento. Las dudas se preguntan exclusivamente al profesor ya que son de interés general.

- El objetivo de la práctica es desarrollar una subrutina en lenguaje ensamblador equivalente a una función de C dentro de una aplicaciónd de generación de imágenes con formato BMP.
- Los cuatro primeros ejercicios en lenguaje C se realizarán de forma guiada con el profesor y el resto de forma **individual**.

2. Formato BMP

2.1. Codificación

- El formato BitMapPortable (BPM) es un formato de imagen escalar, es decir, contiene los datos de cada pixel codificando la intensidad de los componentes RGB de color tal como se visualizará en la pantalla.
- La pantalla está formada por una matriz bidimensional de pixeles, donde cada pixel es un punto discreto de la pantalla programable. La matriz de la pantalla está vinculada a una estructura de datos tipo array bidimensional 2D de filas (eje horizontal) y columnas (eje vertical) almacenada en la memoria de la tarjeta de video. El origen de coordenadas del array es la esquina inferior izquierda. A cada par (x,y) del array 2D le corresponde el color de un pixel.
- True Color: cada elemento del array contiene un dato formada por 3 campos, donde cada campo representa un color (Blue-Green-Red) y ocupa un byte . Cada componente de color R-G-B está codificado con un byte que indica la intensidad del color. Ejemplos:
  - R-G-B:0xFF-0x00-0x00 → pixel 100 % rojo e intensidad máxima.
  - R-G-B:0xFF-0x00-0xFF → pixel 50 % rojo y 50 % azul → color morado.
  - R-G-B:0x00-0x00-0x00 → ausencia de color → color negro
  - R-G-B:0xFF-0xFF-0xFF → misma proporción de colores primarios → color blanco
  - R-G-B:0x7F-0x7F-0x7F → misma proporción de colores primarios → escala de grises entre el negro (00-00-00) y el blanco (FF-FF-FF)
- Una imagen de tamaño en pixeles 512x512 dara lugar a un array de 512 pixeles x 512 pixeles x 3 bytes/pixel

2.2. Mapa de memoria

- Al escribir los colores del array2D MxN en la memoria lineal donde cada dirección es un byte, la estructura de datos o buffer se escribe de la siguiente forma:
  - F0C0BGR-F0C1BGR-. . . -F0C_(N-1)BGR-F1C0BGR-. . . -F1C_(N-1)BGR-. . . .-F_(M-1)C0BGR-F_(M-1)C1BGR-. . . -F_(M-1)C_(N-1)BGR que se corresponden con las posiciones relativas 0-1-2-3-4-5-. . . ..-(MxNx3-1)
  - F0C0BGR: Fila cero Columna cero Azul Verde Rojo
    - 3 bytes en el orden azul-verde-rojo.
    - longitud del buffer: MxNx3 bytes
    - El byte azul ocupará dentro del buffer la posición relativa u offset 0 , el verde la posición 1 y el rojo la posición 3.
  - F0C1BGR: byte azul → posición 3 dentro del buffer
  - F0C_(N-1)BGR: byte azul → posición 3*(N-1)
  - F1C0BGR: byte azul → 3*N
  - F1C_(N-1)BGR: byte azul → 3*N+3*(N-1)
  - F_iC_jBGR: byte azul → 3*N*i+3*j donde 0⇐i<M y 0⇐j<N

2.3. Fichero

- Las imágenes con formato BMP se guardan en ficheros con extensión "*.bmp" como "test.bmp"
- El fichero BMP además del buffer de datos contiene una cabecera con metainformación que no procede explicar en este contexto.

3. Módulo Fuente bitmap_gen_test.c

3.1. Descripción

- Genera un array de pixeles y lo salva en el fichero test.bmp

3.2. Funciones

3.2.1. main()

- Descripción de bloques

```
RGB_data buffer[512][512]  : variable local done se declara y genera el array 2D "buffer"  ↩
    de pixeles donde cada pixel es del tipo RGB_data

Tipo RGB_data:  3 bytes consecutivos donde el primero es la intensidad de azul, el segundo ↩
     verde y el tercero rojo. Las intensidades son números enteros sin signo.
```

3.2.2. memset(buffer, 0, sizeof(buffer))

- inicializa a cero el array 2D de pixeles "buffer"

3.2.3. bmp_generator("./test.bmp", 512, 512, (BYTE*)buffer)

- Genera el fichero "test.bmp" y escribe en dicho fichero el contenido del array 2D de pixeles buffer.

3.2.4. bucle doble

- bucle **for** :
  - la variable i es el índice de filas y la variable j el índice de columnas.
  - buffer[i][j].b : byte blue del pixel de la posición (i,j)
  - buffer[i][j].g : byte green del pixel de la posición (i,j)
  - buffer[i][j].r : byte red del pixel de la posición (i,j)

4. Ejercicios

4.1. Programación en C

- Leer el procedimiento de programación en el fichero **LEEME.txt**
- El objetivo es modificar la función principal **main()** del programa original **bitmap_gen_test.c** dando lugar a distintos programas independientes entre sí.

1. - Compilar y ejecutar el program *bitmap_gen_test.c*
2. - visualizar la imagen del fichero test.bmp: **$display test.bmp**
3. - Módulo **cuadrado_128x128.c** :Cambiar las dimensiones de la imagen a 128 pixeles x 128 pixeles editando la macro DIMENSION=128 un gris con una intensidad del 50% de su valor máximo.
4. - Módulo **cuadrados_4.c**: Generar 4 cuadrados, uno dentro de otro simétricamente, donde el cuadrado mayor negro es 512x512 y el resto se reduce 1/8 cada uno. No utilizar ctes en las sentencias de C, utilizar las macros x_coor, y_coor, top para indicar el valor inicial del **for** y la posición máxima (top) de las filas y columnas. Colores de los cuadrados: background (00-00-00)/(FF-00-FF)/(00-FF-FF)/(FF-FF-00)/
5. - Módulo **bmp_funcion.c**: El bloque de código que realiza el bucle para inicializar los pixeles del cuadrado convertirlo en la función:
   - prototipo: *void pixels_generator(unsigned int x, unsigned int y, unsigned int maximo, RGB_data reg_mem[][top])*
     - x e y son el origen de coordenadas del cuadrado
     - maximo es la coordenada mayor del cuadrado
     - llamada a la función: pixels_generator(xcoor,ycoor,top,buffer);
       - los argumentos xcoor=top/8, ycoor=top/8 y top=512 definirlos mediante macros

4.2. Programación en ASM

1. - Módulo **bmp_as.c**: Implementar la función *void pixels_generator(unsigned int maximo, RGB_data reg_mem[][top])* desarrollando en lenguaje ensamblador la subrutina *pixels_generator* en el nuevo fichero **array_pixel.s**. El fichero en lenguaje ensamblador únicamente contendrá la subrutina.
   - La subrutina implementa el doble bucle.
   - De forma implicita en la propia subrutina consideraremos los argumentos x=y=0.
   - Azul, rojo y verde son las intensidades de todos los pixeles del cuadrado.

4.3. GDB

1. En el programa en **bmp_funcion.c** indicar la posición de la pila donde se salva la dirección de retorno de la subrutina **pixels_generator**, así como el contenido del frame pointer y del stack pointer.
2. Lo mismo que en el apartado anterior con el programa **bmp_as.c** para la subrutina *pixels_generator*