Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
IE0217 – Estructuras abstractas de datos y algoritmos para ingeinería

Propuesta proyecto 0
Grupo 04

Fabián Guerra Esquivel, B53207
Yeison Rodríguez Pacheco, B56074

12/10/18

Índice

1. Introducción

El conocimiento de las estructuras que almacenan imagenes y el procesamiento de las mismas es una habilidad que es de gran utilidad en estas épocas, ya que cada vez las cámaras son más baratas y hay más poder de procesamiento, por lo que es de esperarse que se de un auge en los algoritmos relacionados con el procesamiento digital de imagenes y por lo tanto mayor aparición en el mercado laboral.

Para trabajar con imagenes se debe conocer cómo es su funcionamiento en un computador. Para una imagen a color RGB en cada pyxel existen tres bytes, uno para cada componente de color (rojo, verde y azul), los cuales varían de 0 (sin intensidad) a 255 (toda la intensidad), por lo que con este simple funcionamiento podemos representar cerca de 16 millones de tipos de colores distintos en un ordenador. Para almacenar estos bytes existen muchos tipos de estructuras, algunos realizan arreglos tridimensionales y otros prefieren trabajar en arreglos bidimiensionales. [1]

En este proyecto se va a realizar un programa que sea capaz de crear imagenes a partir de un archivo binario, del cual se extraerán los bytes, se debe crear una estructura de datos para almacenar los mismos y utilizar una biblioteca para generar la imagen.

2. Reseña del algoritmo

Para resolver este problema primero se debe realizar la lectura de un archivo binario y guardar en bytes los valores del mismo, esto para producir más adelante una imagen a color. También se necesita averiguar el peso del archivo binario, esto con el fin de que el programa haga los cálculos de cuales son las dimensiones del arrelgo que va a guardar los datos, y la cantidad de memoria dinámica que se debe reservar.

Para guardar los datos de los bytes se usará una matriz tridimensional de unsigned char, ya que este tiene un valor entre 0 y 255, al igual que los colores en RGB, por lo que se maximizará el ahorro de la memoria, ya que los binarios pueden llegar a ser grandes en tamaño.

Por último se usará ImageMagick para la creación de la imagen, y se utilizará la biblioteca libpng para conversión entre formatos.

Todo el programa se desarrollará y experimentará utilizando linux, para la programación del código se utilizará Visual Studio Code, para la compilación se hará uso de la herramienta make, para el control de versiones se utilizará GitLab y para la documentación se utilizará doxygen.

3. Funcionamiento del algoritmo

3.1. Lectura archivo binario

Primeramente en el algoritmo se deberán leer bytes desde un archivo binario, para esto se utilizarán funciones de lectura de archivos que contiene el lenguaje C++. Para hacer el manejo de un fichero en este lenguaje se utiliza la función fopen('ruta del archivo', 'modo de uso'), la cual recibe como primer parámetro la ruta del archivo que deseamos operar, y como segundo parámetro el modo de uso, que es cómo se va a usar dicho archivo (lectura, escritura, etc). En nuestro caso solo se hará lectura del archivo binario para extraer los bytes del mismo, por lo que se usaría la etiqueta 'r'. Esta función retorna un puntero a un fichero que deberá ser almacenado en una variable, para el uso posterior del mismo. Para la lectura del archivo binario se utiliza la función fread(void *ptr,size_t

size,size_t nitems,FILE *stream) en la cual se reciben cuatro parámetros de entrada, void *ptr es un puntero que apunta a donde se van a guardar los datos leídos del archivo, size_t size es el tamaño en bytes de los datos que se van a leer, en nuestro caso sería un byte, ya que es como se representan los colores. size_t nitems es la cantidad de elementos que se desean leer y FILE *stream es el puntero al archivo creado con la función fopen. [2]

3.2. Tamaño de la imagen

Para crear una imagen a partir del archivo binario, es necesario saber de qué tamaño será creada. Al hacer una imagen cuadrada o rectangular, es posible que vaya a haber pérdidas de bytes, debido a que el tamaño del archivo no necesariamente va a calzar con la cantidad de bytes necesarios para completar la imagen. Para abordar este problema y averiguar el tamaño de la imagen que se quiere hacer, se tomarán formas distintas de imágenes: cuadrada rectangulares con diferentes proporciones entre sus lados. Así, se creará un algoritmo que identifique cuál de las formas hace que se adapte mejor el tamaño del archivo con el de la imagen. Por ejemplo, si se tiene un archivo binario que resulta en 30 pixeles, y se tienen tres formas: cuadrada, rectangular 2:3 y rectangular 1:3, la cuadrada quedaría de 5x5 pixeles, la 2:3 quedaría de 4x6 pixeles y la de 1:3, de 3x9 pixeles, por lo que se eligiría la tercera opción, debido a que se pierde la menor cantidad de pixeles al crear la imagen.

3.3. Lectura del archivo y almacenamiento de datos

Para obtener los valores requeridos del archivo binario, simplemente se abrirá cualquier archivo binario en forma de lectura. Debido a que el formato de las imágenes a color RGB tienen 3 componentes, se creará una matriz Alto * Ancho * 3, es decir, cada coordenada en (x,y) tendrá los tres valores RGB necesarios.

3.4. Creación de la imagen

Para crear la imagen en el programa, una vez teniendo el tamaño de la imagen, se utilizará código ya existente de software o bibliotecas para crear las imágenes en los distintos formatos.
Entre los software más utilizados para la creación de imágenes y convertidores de formato, está ImageMagick. Además, se puede utilizar la biblioteca de C++ *libpng*, la cual es una biblioteca con una amplia gama de funciones para visualizar, convertir formato y editar imágenes. [3]

4. Experimentos que se realizarán

Primero se realizarán los experimentos correspondientes a la lectura de archivos binarios, para saber si el algoritmo utilizado para extraer los bytes no presenta errores, probando con archivos de diferentes tamaños y viendo como el programa reacciona ante diferentes dimensiones de imagenes (ya que no todas van a ser cuadradas).

Con respecto a la estructura tridimensional de unsigned char, se experimentará con la misma agregando y operando valores de manera manual, para observar su correcto funcionamiento. Seguidamente se conectará esta etapa con la de los bytes leídos del archivo binario y se experimentará con distintos archivos para verificar si existen errores y así corregirlos. Cuando se tenga la estructura funcionando junto con la lectura del archivo binario se procederá a crear las imagenes utilizando

ImageMagic, se probará nuevamente generando con distintos binarios la gereración de la imagen, y también se debe verificar que funcione para distintos formatos.

5. Conclusiones

- Para resolver este problema se decidió dividirlo en pequeñas secciones para así trabajar cada una de manera profunda y evitar posibles errores. Hasta que una sección esté terminada y probada no se continuará con la otra.
- Ya se cuenta con la biblioteca y el programa que generará las imagenes, por lo que se pudo adecuar la estructura del arreglo de bytes al mismo, para su correcto funcionamiento.
- Se contará con una función que verifique cuál es el tamaño de la imagen óptimo para aprovechar la máxima cantidad de bytes posibles del binario.
- Se utilizará la memoria exacta para que el arreglo funcione correctamente, por lo que se optimizará el uso de este recurso

Referencias

[1] Yuan,F. (2001). *Windows Graphics Programming: Win32 GDI and DirectDraw*. Prentice Hall PTR. 430.

[2] Didact,I. (2005). *Manual de programacion lenguaje c++*. MAD. 71-76.

[3] libpng (2018). *libpng*. Extraído el 12 de octubre del 2018 de *http://www.libpng.org/pub/png/libpng.html*.