Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingenierías y Tecnologías Avanzadas

Practica 6: Formato BMP

Violeta

Alumno: Hernandez Lomeli Nestor

Boleta: 2024640029

Asignatura: Multimedia

Código formato BMP pero en colores violeta

```
# Binary filter: 16 Colors

file = open('./Images/volcan.bmp','rb')
fileo = open('./Images/violetabin.bmp','wb')
metadata = file.read(54)
fileo.write(metadata)

# --- TUS COLORES ---
violeta1 = [0xFF,0xD4,0xEA]
violeta2 = [0xFF,0xAA,0x4D]
violeta3 = [0xFF,0x80,0xFB]
violeta4 = [0xFF,0x55,0xAA]
violeta5 = [0xFF,0xA2,0x59]
violeta6 = [0xFF,0x00,0xF7]
violeta7 = [0x5D,0x00,0xA6]
violeta8 = [0xAA,0x00,0x55]
violeta9 = [0x08,0x00,0x04]
violeta10= [0x55,0x00,0xA2]
violeta11= [0xB2,0x00,0x51]
violeta12 =[0xFF,0x00,0x08]
violeta13 =[0xFF,0x00,0xF6]
violeta14= [0xFF,0x00,0xF8]
violeta15= [0x2E,0xB2,0xA8]
violeta16= [0xFF,0xD4,0xEA]

# --- CORRECCIÓN 1: AGRUPAR ---
# Metemos tus variables en una lista para poder acceder por número (0 al
# El orden importa: celeste1 será el índice 0, celeste16 el índice 15.
```

```
paleta = [
    violeta1, violeta2, violeta3, violeta4, violeta5, violeta6,
    violeta7, violeta8, violeta9, violeta10, violeta11,
    violeta12, violeta13, violeta14, violeta15, violeta16
]

file.seek(54,0)
no_pix = 0

# --- CORRECCIÓN 2: EL LÍMITE ---
# Esto está perfecto. Es el tamaño de cada "rebanada" del pastel.
limite = (pow(2, 24)-1)/16

while(True):
    # --- CORRECCIÓN 3: LEER EL PÍXEL ---
    # Te faltaba leer los 3 bytes del píxel actual
    pixel_data = file.read(3)

    if(len(pixel_data) > 0):
        # Convertimos los 3 bytes a un número gigante
        valor_int = int.from_bytes(bytes(pixel_data), byteorder='little'

        # --- CORRECCIÓN 4: CÁLCULO DEL ÍNDICE ---
        # En lugar de usar 'match', calculamos qué posición de la lista
        # Dividimos el valor del píxel entre el tamaño del límite.
        indice = int(valor_int / limite)

        # Seguridad: Si el cálculo da 16 (por ser el máximo valor posibl
        if indice > 15:
            indice = 15
```

```
        # Seguridad: Si el cálculo da 16 (por ser el máximo valor posib
        if indice > 15:
            indice = 15

        # --- CORRECCIÓN 5: ESCRIBIR ---
        # Usamos el 'indice' para sacar el color correcto de la lista '
        fileo.write(bytes(paleta[indice]))

        no_pix += 1
    else:
        break

print('No Pixels: '+str(no_pix))
file.close()
fileo.close()
```

El código implementa un filtro binario que reduce los colores de una imagen BMP a una paleta limitada de 16 tonos definidos previamente, en este caso colores violetas. Su objetivo principal es procesar cada píxel de la imagen original y reemplazar su color por uno de los colores de la paleta, generando una nueva imagen con menor variedad cromática y un efecto visual uniforme.

Primero se abren dos archivos: la imagen original en modo lectura binaria y un nuevo archivo en modo escritura binaria donde se guardará la imagen resultante. Posteriormente se leen los primeros 54 bytes del archivo original, que corresponden a la cabecera del formato BMP. Esta cabecera contiene información importante como el tamaño de la imagen, la resolución y la profundidad de color. Dichos bytes se copian directamente al archivo de salida para conservar la estructura del formato y asegurar que la nueva imagen pueda visualizarse correctamente.

Después se definen 16 colores en formato hexadecimal que conforman la paleta de salida. Cada color se representa como un arreglo de tres valores correspondientes a los canales RGB. Estos colores se agrupan en una lista llamada “paleta”, lo que permite acceder a ellos mediante un índice numérico del 0 al 15 durante el procesamiento de los píxeles.

El programa posiciona el puntero del archivo justo después de la cabecera para comenzar a leer los datos de la imagen. También se inicializa un contador para registrar el número de píxeles procesados. Luego se calcula un valor llamado “límite”, el cual se obtiene dividiendo el número total de combinaciones posibles de color en una imagen de 24 bits 2^{24} entre 16. Este valor sirve para dividir el espacio de colores en 16 rangos iguales, de manera que cada rango corresponda a uno de los colores de la paleta.

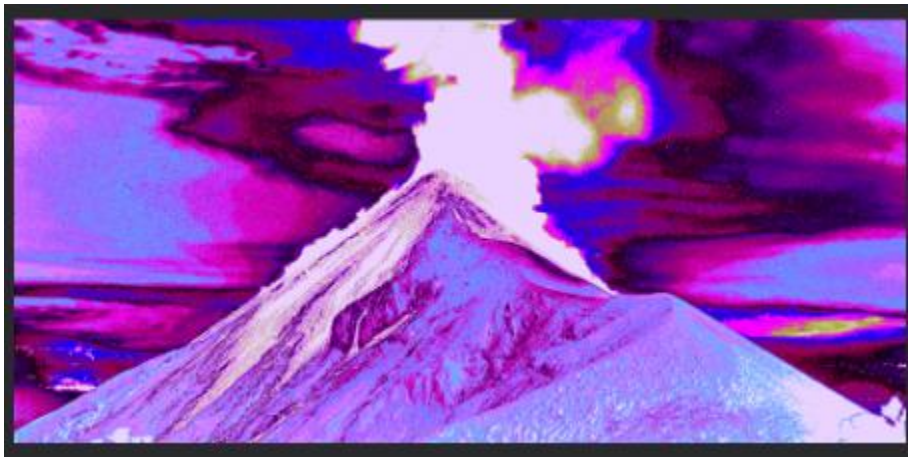
A continuación, el algoritmo entra en un ciclo donde se leen tres bytes por cada píxel, ya que una imagen BMP de 24 bits almacena cada color mediante tres componentes. Estos bytes se convierten en un número entero para facilitar su comparación. El valor obtenido se divide entre el límite calculado previamente para determinar a qué rango pertenece el píxel. El resultado de esta operación se usa como índice para seleccionar el color correspondiente dentro de la paleta. Si el índice supera el valor máximo permitido, se ajusta para evitar errores.

Una vez seleccionado el color, se escribe en el archivo de salida reemplazando el color original del píxel. Este proceso se repite hasta que se han leído todos los datos

de la imagen. Finalmente, se imprime en pantalla la cantidad total de píxeles procesados y se cierran los archivos para finalizar la ejecución.

En términos generales, el programa realiza una cuantización de color, es decir, reduce la cantidad de colores disponibles en la imagen original a un conjunto limitado de 16 tonos. Esto permite simplificar la información visual, generar un efecto estilizado en la imagen y comprender el funcionamiento básico del procesamiento digital de imágenes a nivel de píxel.

Resultado Final del código



Aquí podemos verificar que efectivamente el código utilizo los 16 tonos de violeta para la imagen y así es el resultado final de este código, cada pixel tiene su propio color de violeta.