Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Cpomputación

IIC2233 Programación Avanzada (2019-1)

Tarea 03: PNG y Filtros

1. Manejo de *bytes*: formato PNG

Portable Network Graphics (PNG) es un formato gráfico que permite almacenar información de imágenes en *bytes*; y todos están en formato *big endian*[1]. Este formato de archivo **siempre** comienza con una firma que nos indica que es un archivo PNG, formada por 8 *bytes* cuyos valores decimales son:

```
137 80 78 71 13 10 26 10
```

Luego, se divide su contenido en bloques de *bytes* (*chunks*), los cuales nos entregarán toda la información sobre la imagen (véase Figura 1).

1	2	3	4	5	6	7	8	9	10	11	12	13	...	k	...	n	$n+1$	$n+2$	...	$n+h$	...
Firma formato PNG								*chunk*							...	*chunk*					...

Figura 1: Organización global en *bytes* del formato PNG para una imagen.

Cada uno de estos *chunks* posee 4 secciones ordenadas:

1. **Largo:** Los primeros 4 *bytes* indican el largo del contenido (sección información) del bloque.
2. **Tipo:** Son los siguientes 4 *bytes*, que se pueden decodificar en un *string* de 4 caracteres ASCII. Indican el tipo de información que va a contener el *chunk*. Hay muchos tipos de *chunks*, pero para que una imagen se pueda ver correctamente, sólo necesitas poder leer 3 tipos: `IHDR`, `IDAT` y `IEND`. Los demás tipos puedes ignorarlos.
3. **Información:** Luego viene el contenido del *chunk*, que depende del tipo.
   - Para `IHDR`: este *chunk* tiene *metadata* de la imagen completa, como las dimensiones, tipos de colores, etc...
   - Para `IDAT`: este *chunk* posee los *bytes* de la comprensión de los píxeles de la imagen. En una imagen pueden existir varios bloques `IDAT`. En este caso particular, se debe generar un solo bloque `IDAT`.
   - Para `IEND`: este *chunk* indica el término del archivo, y en la sección información no posee datos (el largo de la sección es 0).
4. **CRC:** Los últimos 4 *bytes* del *chunk* corresponden a un código CRC (*Cyclic Redundancy Check*), que asegura la integridad de la información del bloque completo. El CRC se calcula con todos los *bytes* del bloque respectivo, sin contar con la sección de largo. Es decir, la sección del tipo de bloque,

[1] Para transformar `integers` a *bytes* en formato *big endian*, se recomienda usar los métodos `int.to_bytes(number, length, 'big')` e `int.from_bytes(bytes, 'big')`

más la sección de datos del bloque. Para calcularlo, debes usar el método `crc32` de la librería `zlib`. El CRC siempre debe estar presente, incluso si el bloque no tuviese información.

Chunk	
Cantidad de *bytes*	Descripción
4	Largo de información de bloque
4	Tipo de bloque
Largo	Información o contenido
4	CRC

Figura 2: Estructura de un *chunk*.

1.1. Tipos de *chunks*

1.1.1. `IHDR`

Es el que contiene los datos principales de la imagen, los cuales son:

- **Ancho**: cantidad de píxeles de ancho. Usa 4 *bytes*
- **Alto**: cantidad de píxeles de alto. Usa 4 *bytes*
- **Profundidad en *bits***: es la cantidad máxima de bits por canal. Generalmente se usa el valor 8, lo que significa que cada canal tendrá valor máximo $2^8 \rightarrow 255$. Usa 1 *byte*.
- **Tipo de colores**: para efectos de esta tarea, puede tener valor 2 o 6 (cualquier otro tipo no se usará), y usa 1 *byte*.
  Si es de valor 2, cada píxel será una tripleta de *bytes*, representando cada uno a la intensidad de cada canal de color RGB (*Red - Green - Blue*), que va desde 0 hasta 255.
  Si es de valor 6, cada píxel será la misma tripleta RGB, más un valor *alpha*, que representa la *opacidad* de la imagen (0 corresponde a transparencia 100 %, mientras que 255 corresponde a transparencia 0 %).
- **Tipo de compresión**: es el algoritmo que se utiliza para comprimir los datos de la imagen, presentes en la información del `IDAT`, y siempre toma el valor 0. Usa 1 *byte*.
- **Tipo de Filtro**: indica el método de preprocesamiento que se utilizó para comprimir los datos, mejora la compresión. Para efectos de esta tarea no se utilizará filtro, entonces siempre llevará el valor 0. Utiliza 1 *byte*.
- **Tipo de entrelazado**: indica el orden de transmisión de datos de la imagen. Para esta tarea no se utilizará entrelazado, entonces siempre llevará el valor 0. Utiliza 1 *byte*

IHDR	
Cantidad de *bytes*	Descripción
4	Ancho en píxeles
4	Alto en píxeles
1	Profundidad de *bits*
1	Tipo de colores
1	Tipo de compresión
1	Tipo de filtro
1	Tipo de entrelazado

Figura 3: Estructura de `IHDR`.

1.1.2. `IDAT`

El bloque de información de `IDAT` es una representación de la matriz de píxeles de la imagen. La información está comprimida mediante el algoritmo de compresión de PNG. Para comprimir, se recomienda utilizar el método `compress` de la librería `zlib`.

Como se dijo, la información descomprimida del bloque `IDAT` corresponde a una representación de la matriz de píxeles de la imagen. Para ejemplificar la construcción de este bloque:

1. Cada *byte* corresponde a un canal de color del píxel que se está representando.
   - El primer *byte* representa el valor del canal rojo del primer píxel;
   - el segundo *byte*, el canal verde del primer píxel;
   - el tercer *byte*, el canal azul del primer píxel;
   - el cuarto *byte*, el canal rojo del segundo píxel, y así sucesivamente
2. Además, cada fila de la matriz de píxeles comienza con un *byte* que representa el filtro aplicado a dicha fila. Como la imagen no tiene filtro, éste debe ser un 0. Por ejemplo, si cada píxel está representado por un triple RGB, el largo de cada fila de la matriz será $1 + \texttt{ancho} \times 3$. Por lo tanto, el largo total del bloque de información del `IDAT` será $\texttt{alto} \times (1 + \texttt{ancho} \times 3)$. Donde `ancho` y `alto` son el ancho y el alto de la imagen en píxeles, respectivamente.
3. Finalmente, se debe comprimir el bloque de información.

1.1.3. `IEND`

Este bloque es el que va al final del archivo. No contiene información, aunque igual **mantiene** la estructura de *chunks*.

1.2. Convertir una matriz de píxeles RGB a una imagen PNG.

La matriz de píxeles RGB representa el color de cada píxel dentro de la matriz, a través de tres valores en una tupla RGB (*Red - Green - Blue*) que simbolizan la composición del color, en términos de la intensidad de los colores primarios aditivos de la luz. Ante cualquier duda respecto a este proceso, y en particular, cómo construir los bloques, se recomienda revisar la especificación de PNG, ubicada en el siguiente enlace.

Supongamos que queremos crear una imagen PNG de 4×3 píxeles (4 de ancho y 3 de alto), sin filtro ni entrelazado (como ya se mencionó, no trabajaremos con ninguno de estos en la tarea). Construimos las partes de la imagen en orden:

1. Crear el *header*. Para esto, debemos codificar en un *byte* cada uno de los números de la secuencia entregada en Manejo de *bytes*: formato PNG, y luego concatenarlos para formar el inicio del contenido de la imagen.

2. Crear el `IHDR`. Seteamos el tipo de *chunk*: `tipo = "IHDR"` $\rightarrow$ 4 *bytes*.

   Luego, especificamos *metadata* de la imagen:

   - `ancho = 4` $\rightarrow$ 4 *bytes*
   - `alto = 3` $\rightarrow$ 4 *bytes*
   - `profundidad = 8` $\rightarrow$ 1 *byte*
   - `color = 2` $\rightarrow$ 1 *bytes*
   - `compresion = 0` $\rightarrow$ 1 *byte*
   - `filtro = 0` $\rightarrow$ 1 *byte*
   - `entrelazado = 0` $\rightarrow$ 1 *byte*

   Luego, concatenamos toda esta información como *bytes*, digamos en la secuencia `información` y calculamos su largo:

   $$\texttt{largo(informacion) = 4 + 4 + 1 + 1 + 1 + 1 + 1 = 13} \rightarrow \text{ 4 } \textit{bytes}$$

   Ahora, calculamos el CRC. Para hacer esto, concatenamos `tipo` y `información`, y usamos el método `crc32` para convertir. Guardamos este resultado en 4 *bytes*. Luego, no queda nada más que unir todo:

   $$\texttt{chunk = largo + tipo + informacion + crc}$$

3. Crear el `IDAT`. Fijamos el tipo de *chunk*: `tipo = ÏDAT"` $\rightarrow$ 4 *bytes*.

   La información será todos los píxeles unidos de cada fila (cada uno en 1 *byte*), más un *byte* (con valor 0, porque no tiene filtro) al comienzo de cada una de ellas:

   $$\text{fila}_i = 0 \; r_{i,1} \; g_{i,1} \; b_{i,1} \; r_{i,2} \; g_{i,2} \; \ldots$$

   Donde $r_{i,j}$ corresponde al valor del canal rojo para el píxel en la fila i y la columna j de la imagen (y respectivamente $g_{i,j}$ y $b_{i,j}$ con verde y azul). Luego, se concatenan todas las filas y se comprime la información usando el método `compress`. Calculamos el largo y el CRC al igual que el punto anterior, finalmente unimos toda la información.

4. Crear el `IEND`. Se realiza de la misma forma que se crearon los otros dos *chunks*, considerando que el largo de información de este bloque es 0, pero aún así debe llevar CRC.

2. Filtro Dibujo

La aplicación de este filtro requiere que realices varias operaciones matriciales sucesivas a la matriz de píxeles RGB, para que luego sea guardada y pueda ser abierta por tu ordenador. Para lograr esto, debes implementar las siguientes funciones:

2.1. Transformación de una imagen de color a escala de grises.

En tratamiento de imágenes se suele utilizar imágenes en escala de grises. El procedimiento consiste en efectuar una reducción de los planos de color a tonos grises. Un modelo que realiza esta transformación es:

$$\texttt{X} = 0.299 \cdot \texttt{R} + 0.587 \cdot \texttt{G} + 0.114 \cdot \texttt{B}$$

donde `X` es el valor en tonos de grises, `R` es el valor correspondiente al rojo, `G`, al verde, y `B` al azul, para un píxel cualquiera.

***Hint*:** El formato PNG requiere que cada píxel tenga los 3 canales de colores. Un píxel de la forma `[x, x, x]` tiene intensidad `x` en escala de grises. Sin embargo, se recomienda mantener la matriz con solo 1 canal para realizar las siguientes operaciones. En el caso de que se quiera guardar el canal *alpha* se puede setear en 255, para que la imagen no sea trasparente.

2.2. Binarización de una imagen en escala de grises

Una imagen en escala de grises se puede transformar en una imagen binaria, esta es, una imagen cuyos píxeles sólo pueden estar "encencidos" o "apagados". Para ello, se fija una `cota`, tal que todos los píxeles cuya intensidad es inferior a ese valor, tomarán valor `0`, y todos los píxeles cuya intensidad es superior se fijan en `255`. El valor de la `cota` para esta tarea será 30.

2.3. Inversión de una imagen binaria

Este procedimiento consiste en recibir una imagen binaria (cuyos píxeles tienen intensidad `0` ó `255`), e invierte el valor de cada uno de sus píxeles. Es decir, los píxeles que tienen valor `0` pasarán a tomar valor `255`, y los píxeles con valor `255` pasarán a tener valor `0`.

2.4. Matriz gradiente

La matriz **gradiente** es el resultado de la aplicación de distintas operaciones. Para esto, se debe definir el **Producto de Convolución.**

2.4.1. Convolución de matrices

La operación fundamental para operar con la vecindad de un píxel cualquiera (es decir, los que están a su alrededor), se llama convolución. Esta consiste en una manera de multiplicar dos matrices de diferentes tamaños. Generalmente, la segunda matriz (llamada **núcleo**) será de tamaño mucho menor que la primera (la imagen). Si la imagen tiene M filas y N columnas, y el núcleo tiene m filas y n columnas, entonces la matriz resultante del producto de convolución tendrá $M - m + 1$ filas y $N - n + 1$ columnas.

La convolución deslizará el núcleo sobre la imagen, moviéndolo a través de todas las posiciones de la imagen (con la restricción de que siempre debe quedar **completamente contenido** dentro de la imagen). En cada posición del núcleo, se multiplicarán los valores del núcleo por los valores de la imagen sobre los que esté el núcleo. Luego, se suman todos los valores resultantes. A modo de ejemplo:

$$\text{Sea } I = \begin{bmatrix} 1 & 8 & 3 & 7 \\ 0 & 10 & 15 & 8 \\ 7 & 8 & 4 & 6 \\ 3 & 2 & 21 & 5 \\ 7 & 61 & 12 & 10 \end{bmatrix} \text{ y } K = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Entonces, el producto de convolución entre I y K es:

$$O = I * K = \begin{bmatrix} 166 & 198 \\ 157 & 152 \\ 184 & 177 \\ 475 & 423 \end{bmatrix}$$

donde, por ejemplo:

$$\begin{aligned} O_{11} &= I_{11} \cdot K_{11} + I_{12} \cdot K_{12} + I_{13} \cdot K_{13} + I_{21} \cdot K_{21} + I_{22} \cdot K_{22} + I_{23} \cdot K_{23} \\ &= 1 \cdot 1 + 8 \cdot 2 + 3 \cdot 3 + 0 \cdot 4 + 10 \cdot 5 + 15 \cdot 6 \\ &= 166 \end{aligned}$$

$$\begin{aligned} O_{32} &= I_{32} \cdot K_{11} + I_{33} \cdot K_{12} + I_{34} \cdot K_{13} + I_{42} \cdot K_{21} + I_{43} \cdot K_{22} + I_{44} \cdot K_{23} \\ &= 1 \cdot 1 + 8 \cdot 2 + 3 \cdot 3 + 0 \cdot 4 + 10 \cdot 5 + 15 \cdot 6 \\ &= 177 \end{aligned}$$

2.4.2. Obtención de la matriz gradiente

Finalmente, para obtener la matriz *gradiente*, debes utilizar 2 núcleos de convolución particulares:

$$M_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \text{ y } M_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Así, a partir de una imagen dada, se deben realizar las siguientes operaciones:

- Se le aplican 2 operaciones de convolución (por separado), una con cada núcleo, resultando en las matrices C_1 y C_2, respectivamente.
- Cada una de estas dos matrices se eleva al cuadrado (elemento a elemento), resultando en las matrices D_1 y D_2, tales que para cada i, j:

$$D_1[i,j] = C_1[i,j]^2$$

$$D_2[i,j] = C_2[i,j]^2$$

- Finalmente, se define la matriz *gradiente* $G = \sqrt{D_1 + D_2}$ tal que, para cada i, j:

$$G[i,j] = \sqrt{D_1[i,j] + D_2[i,j]}$$

2.5. Aplicación del filtro dibujo

Finalmente, los pasos que debes realizar para aplicar el filtro dibujo son:

1. Leer la imagen y transformarla a matriz RGB, utilizando la función `get_pixels` de la librería entregada `pixel_collector.py`
2. Transformar la imagen a escala de grises
3. Obtener la matriz **gradiente** de esta imagen
4. Binarizar esta matriz **considerando una cota de 30**
5. Invertir esta imagen binaria
6. Convertir la matriz a formato PNG y guardar el resultado