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

IIC1103 – Introducción a la Programación
1 - 2016

Enunciado Tarea 2

Recordatorio:

- **Fecha de entrega:** 27 de mayo del 2016 a las 23:59 hrs.
- **Foro de consulta:** `https://goo.gl/IBpKw2`
- Este trabajo es **estrictamente personal**. Recuerda leer la Política de Integridad Académica del DCC disponible en `http://www.ing.uc.cl/ciencia-de-la-computacion/programas/licenciatura/politica-de-integridad-academica/`. Se usará un software anti-plagio para detectar la copia (similitud entre códigos).

Objetivo

En esta tarea, se espera que practiques los contenidos de control de flujo, ciclos, strings y listas. Para esto, deberás realizar un programa que permita al usuario realizar acciones sobre imágenes digitales.

Procesamiento digital de imágenes

Un *pixel* es la unidad básica de toda imagen digital, y es una abreviación del inglés *picture element*. Así, se puede representar las imágenes digitales mediante una matriz, donde cada celda se denomina un *pixel* (ver Figura 1).

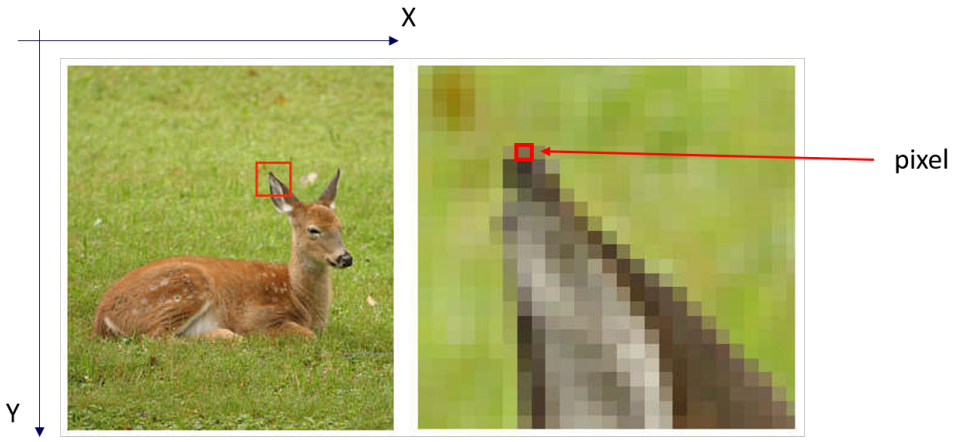


Figura 1: Obtenido de http://photo.net/equipment/digital/basics/

Para imágenes en color, el *pixel* contiene la información sobre la intensidad de los colores primarios de la luz: rojo, verde y azul. Cada pixel tiene valor RGB (por sus siglas en inglés *red*, *blue* y *green*). Este valor está compuesto por tres valores enteros (que van de 0 a 255) de rojo, verde y azul. El valor 0 se interpreta como *'ausencia'* del respectivo color. Así, cada pixel P_{ij} tiene tres componentes: R_{ij}, G_{ij} y Bij. Observa los siguientes ejemplos:

R_{ij}	G_{ij}	B_{ij}	Color del *pixel* P_{ij}
0	0	0	Negro
255	255	255	Blanco
255	0	0	Rojo
0	255	0	Verde
0	0	255	Azul
255	255	0	Amarillo
0	255	255	Cian
255	0	255	Magenta

Gracias a esta representación discreta en pixeles, es posible realizar una serie de operaciones sobre las imágenes, las que se explican a continuación:

Invertir horizontalmente/verticalmente

Se invierten las columnas o las filas, dependiendo si se quiere invertir horizontal o verticalmente. Supongamos que tenemos una imagen de 36 pixeles, para invertir la imagen horizontal o verticalmente se realiza lo siguiente:

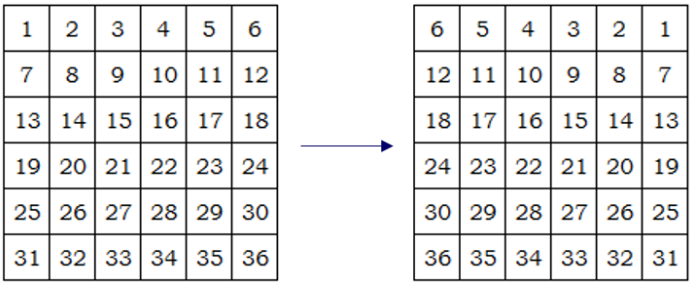


Figura 2: Proceso para invertir horizontalmente

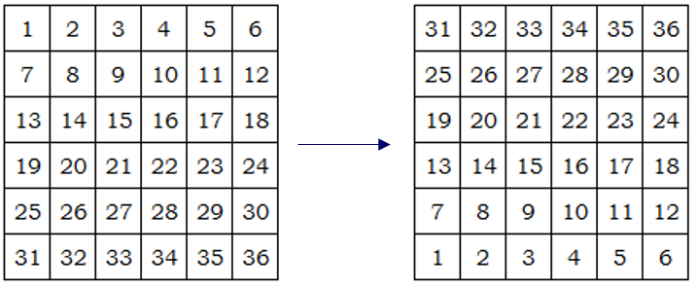


Figura 3: Proceso para invertir verticalmente

Para el ejemplo de imagen original con la que se está trabajando, la imagen invertida horizontal y verticalmente quedará de la siguiente forma:

Imagen original

Imagen invertida verticalmente

Imagen original

Imagen invertida horizontalmente

Sepia

Para realizar esta transformación, se considera que para cada pixel P_{ij}, con componentes (R_{ij}, G_{ij}, B_{ij}) se realiza la siguiente operación, dejando como resultado el pixel P'_{ij}, con componentes $(R'_{ij}, G'_{ij}, B'_{ij})$:

$$R'_{ij} = [0{,}393 * R_{ij} + 0{,}769 * G_{ij} + 0{,}189 * B_{ij}]$$
$$G'_{ij} = [0{,}349 * R_{ij} + 0{,}686 * G_{ij} + 0{,}168 * B_{ij}]$$
$$B'_{ij} = [0 * 272 * R_{ij} + 0{,}534 * G_{ij} + 0{,}131 * B_{ij}]$$

Donde [] representa la función parte entera. Considera que en caso de haber cualquier valor sobre 255, éste tomará el valor 255. Al realizar la transformación *pixel* a *pixel* descrita arriba, la imagen original querdará de la siguiente forma:

Imagen original

Imagen en sepia

Invertir colores

Para realizar esta transformación, se considera que para cada pixel P_{ij}, con componentes (R_{ij}, G_{ij}, B_{ij}) se realiza la siguiente operación, dejando como resultado el pixel P'_{ij}, con componentes $(R'_{ij}, G'_{ij}, B'_{ij})$:

$$R'_{ij} = 255 - R_{ij}$$
$$G'_{ij} = 255 - G_{ij}$$
$$B'_{ij} = 255 - B_{ij}$$

Al realizar la transformación *pixel* a *pixel* descrita arriba, la imagen original querdará de la siguiente forma:

Imagen original

Imagen con colores invertidos

Difuminar

Para difuminar una imagen, debes realizar para cada pixel P_{ij} la siguiente operación: obtener el valor R, G y B de todos los vecinos en todas las direcciones de dicho pixel (8 pixeles) y sumarlos. A esta suma se le agrega el valor R, G y B del pixel P_{ij} multiplicado por 8. Luego, esta suma se divide por 16.
Para ejemplificar, supongamos que quieres realizar la operación para el pixel P_{11} tal como muestra la imagen:

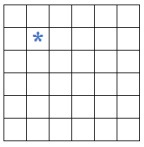


Figura 4: Pixel que se encuentra en la posición $(1, 1)$

Se puede describir este proceso de la siguiente forma:

$$R'_{11} = \frac{R_{00} + R_{01} + R_{02} + R_{10} + R_{12} + R_{20} + R_{21} + R_{22} + 8 * R_{11}}{16}$$

$$G'_{11} = \frac{G_{00} + G_{01} + G_{02} + G_{10} + G_{12} + G_{20} + G_{21} + G_{22} + 8 * G_{11}}{16}$$

$$B'_{11} = \frac{B_{00} + B_{01} + B_{02} + B_{10} + R_{12} + B_{20} + B_{21} + B_{22} + 8 * B_{11}}{16}$$

A veces no será posible obtener los valores R, G y B de todos los vecinos. En este caso, debes considerar todos los vecinos posibles y dividir dicho pixel en $8 + p$ donde p es la cantidad de pixeles vecinos.

Al realizar la transformación pixel a pixel descrita arriba, la imagen original quedará de la siguiente forma:

Figura 5: Imagen original y difuminada

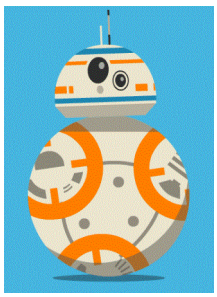
Tu programa

Imágenes

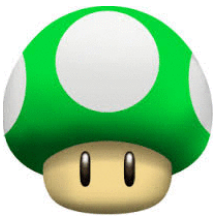
En esta tarea las imágenes estarán guardadas en un archivo de texto. En cada fila se representa cada uno de los pixeles escritos como `(r,g,b)` separados por punto y coma `';'`. Por ejemplo, si se tuviese un archivo que representa una imagen de 4 pixeles de alto y 3 de ancho, el archivo se vería de la siguiente forma:

```
(1, 124, 111);(201, 112, 33);(212, 0, 255)
(10, 12, 10);(21, 55, 73);(21, 30, 55)
(255, 255, 255);(20, 172, 213);(92, 40, 85)
(0, 50, 100);(88, 22, 57);(11, 150, 229)
```

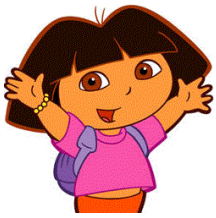
Para efectos de esta tarea, no tendrás que hacer manejo de archivos, pues el módulo que se explica en la siguiente sección se preocupa de ello. Para que practiques en tu tarea, se te entregarán 4 ejemplos, que se muestran a continuación:



Archivo bb8, 447x328 pixeles



Archivo mario, 204x204 pixeles



Archivo dora, 247x249 pixeles

Archivo gato, 199x199 pixeles

Módulo

Para realizar tu tareas contarás con un módulo llamado `Tarea2GUI`. Esta es una librería gráfica que permite dibujar imágenes pixel a pixel en una ventana principal.

Funciones

Las funciones provistas por el módulo se explican a continuación:

1. `cargar_imagen(nombre_archivo)`: recibe el nombre del archivo (sin el `.txt`, por ejemplo `'dora'` o `'mario'`) que representa una imagen. La imagen tiene largo m y ancho n y devuelve una lista de listas de la forma `[[p11,p12,...p1n],[p21,p22,...p2n],...,[ pm1,pm2,...pmn ]]`. Donde cada `pij` es un pixel que se representa como un string de la siguiente forma `'(r, g, b)'` (notar que hay un espacio (`' '`) luego de cada coma. El archivo `txt` debe estar en la misma carpeta que el archivo `.py` de tu tarea y el módulo `Tarea2GUI.py`. Si el nombre del archivo no existe, entonces retorna una lista vacía (`[]`)

2. `guardar_imagen(imagen,nombre)`: recibe una lista de la forma `[[p11,p12,...p1n],[p21,p22,...p2n],...,[ pm1,pm2,...pmn ]]`. Donde cada `pij` es un pixel que se representa como una lista de strings de largo 3: `[r,g,b]` y lo escribe en el archivo `nombre.txt`. Retorna `True` cuando fue posible guardar el archivo.

3. `nueva_ventana(ancho_ventana,largo_ventana)`: inicializa una nueva ventana de ancho `ancho_ventana` y largo `largo_ventana`. No dibuja nada ni despliega la ventana, pero es necesario llamar a esta función antes de llamar las funciones que se enuncian después de ésta.

4. `cambiar_nombre_ventana(nombre_ventana)`: recibe un nuevo nombre para poner como título en la ventana principal.

5. `inicializar_lienzo(alto,ancho)`: crea dos lienzos distintos y vacíos de dimensiones `alto` $\times$ `ancho`. Sobre éstos lienzos se puede pintar uno a uno los pixeles.

6. `dibujar_pixel(num,r,g,b,x,y)`: recibe el número `num` de lienzo sobre el cual se quiere dibujar un pixel. El parámetro `num` puede ser 1 o 2. Además recibe las componentes `r`, `g` y `b` del pixel, cada una de ellas con valores enteros entre 0 y 255 (ambos inclusive). Por último recibe la posición `x` e `y` donde se quiere dibujar el pixel.

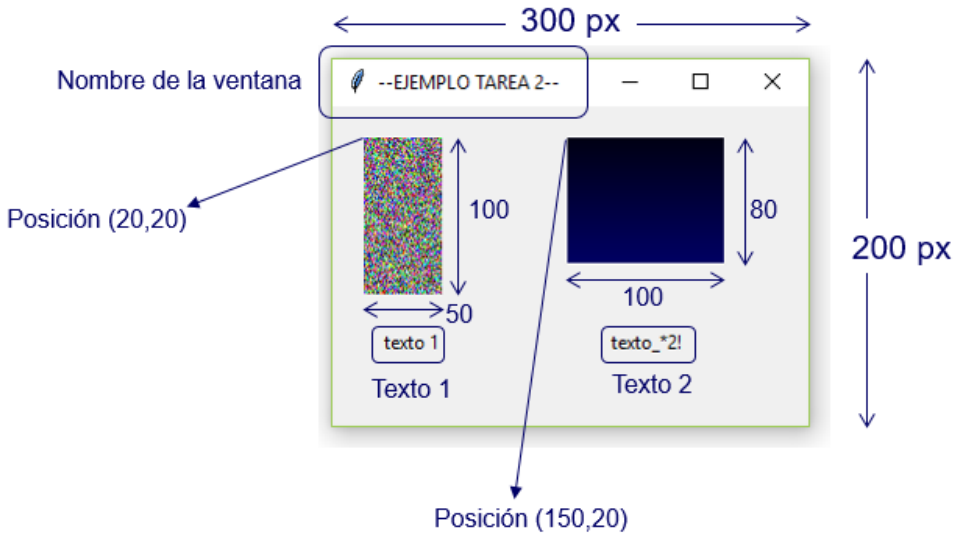
7. `desplegar_imagenes(x_im1,y_im1,x_im2,y_im2,x_t1,y_t1,x_t2,y_t2,t1,t2)`: recibe todos los parámetros necesarios para desplegar ambos lienzos en las posiciones (`x_im1`,`y_im1`) e (`x_im2`,`y_im2`) respectivamente. Además permite agregar dos textos: `t1` y `t2` en las posiciones (`x_t1`,`y_t1`) e (`x_t2`,`y_t2`) respectivamente.

Ejemplo

A continuación, se muestra un código de ejemplo que, al ejecutarse se muestra la siguiente ventana:

```
import random
import Tarea2GUI

Tarea2GUI.nueva_ventana(300,200)
Tarea2GUI.cambiar_nombre_ventana("--EJEMPLO TAREA 2--")
Tarea2GUI.inicializar_lienzo(100,50,80,100)

for y in range(100):
    for x in range(50):
        r = random.randint(0,255)
        g = random.randint(0,255)
        b = random.randint(0,255)

    Tarea2GUI.dibujar_pixel(1,r,g,b,x,y)

b = 20
for y in range(80):
    for x in range(100):
        r = 0
        g = 0
        b = b

        Tarea2GUI.dibujar_pixel(2,r,g,b,x,y)
    b += 1

Tarea2GUI.desplegar_imagenes(20,20,150,20,50,150,200,150,"texto 1","texto_*2!")
print("Se ha cerrado la ventana")
```

Este código genera la siguiente ventana:



Una vez que se cierre la ventana, continuará la ejecución normal del programa y se imprimirá en la consola `"Se ha cerrado la ventana"`

Funcionalidades de tu tarea

Tu tarea debe tener las siguientes funcionalidades:

1. Saludar al usuario
2. Cargar un archivo que exista en la carpeta de tu proyecto, el usuario debe escoger qué archivo quiere cargar
3. Permitir al usuario elegir la acción de transformacin de imagen que desee, entre las siguientes opciones:
   - Invertir horizontalmente la imagen original (imagen 1) y mostrarla junto con la imagen invertida (imagen 2).
   - Invertir verticalmente la imagen original (imagen 1) y mostrarla junto con la imagen invertida (imagen 2).
   - Convertir a sepia la imagen original (imagen 1) y mostrarla junto con la imagen transformada (imagen 2).
   - Invertir los colores de la imagen original (imagen 1) y mostrarla junto con la imagen con sus colores invertidos (imagen 2).
   - Difuminar la imagen original (imagen 1) y mostrarla junto con la imagen difuminada (imagen 2).
4. Guardar la imagen actual, la que corresponde al resultado de la última transformación realizada
5. Ver imagen actual, la que corresponde al resultado de la última transformación realizada
6. Salir

Al realizar todo lo anterior, debes tener en consideración lo siguiente:

- Tu programa debe poder permitir realizar transformaciones sucesivas de una imagen. Al final de una transformación, la imagen final será la imagen original de la siguiente transformación.
- Cuando se abre una ventana con la función `desplegar_imagenes` ésta debe cerrarse para que la interacción con el usuario siga a través de la consola. Para más detalle ve el ejemplo al final de este documento.
- Debes comprobar que tanto los valores de R, G y B que usas en el módulo sean valores enteros entre 0 y 255
- Debes comprobar que las posiciones que uses para poner una imagen o texto esté dentro de la ventana que ha sido creada.

Entrega

Debes guardar tu tarea en un archivo con el formato `tarea02_rut.py`, donde debes reemplazar `rut` con tu RUT. Por ejemplo si tu rut es 12.345.678-9 el nombre de la tarea debería ser `tarea02_123456789.py`. La entrega se realiza en el buzón electrónico, disponible en la página web del curso: `https://puc.classter.net` hasta el 27 de mayo a las 23:59 hrs.

Ejemplo

A continuación se muestra una posible interacción con el usuario:

```
********* BIENVENIDO AL EDITOR DE IMAGENES DE PYTHON *********
En este programa vas a poder abrir, editar y guardar imagenes
************************************************************
* Que deseas hacer?:
* 1. Abrir una imagen
* 2. Editar la imagen actual
* 3. Guardar la imagen actual
* 4. Ver imagen actual
* 5. Salir
* OPCION:  1
Diga el nombre del archivo    dora
Imagen cargada con exito!... Ahora puedes editarla
```

```
************************************************************
* Que deseas hacer?:
* 1. Abrir una imagen
* 2. Editar la imagen actual
* 3. Guardar la imagen actual
* 4. Ver imagen actual
* 5. Salir
* OPCION:  2
* Que edicion quieres realizar?
* 1. Cambiar a sepia
* 2. Difuminar
* 3. Invertir horizontalmente
* 4. Invertir verticalmente
* 5. Invertir colores (negativo)
4
```

Una vez que el usuario ha querido invertir verticalmente la imagen, se debe mostrar una ventana como la siguiente:

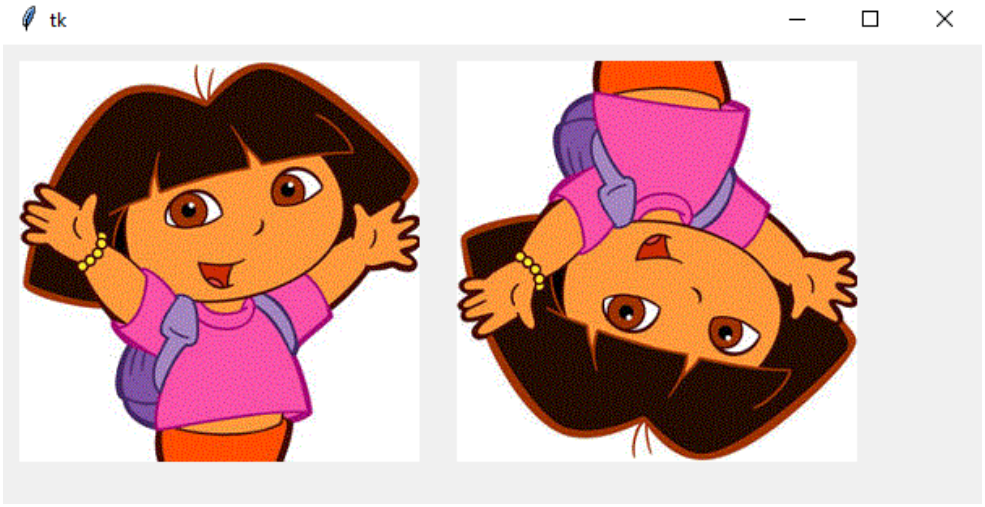


Figura 6:

Cuando el usuario cierre esta ventana, se volverá a mostrar el menú:

```
************************************************************
* Que deseas hacer?:
* 1. Abrir una imagen
* 2. Editar la imagen actual
* 3. Guardar la imagen actual
* 4. Ver imagen actual
* 5. Salir
* OPCION:  5
>> Adios!
```

Hint: Puede ser útil crear funciones para cada una de las transformaciones