UNIVERSIDAD DE SANTIAGO DE CHILE

Simulación de una plataforma

estilo “Photoshop”

Paradigma Orientado a Objetos

Profesor: Miguel Truffa

Estudiante: Nicolás
Gajardo Ponce

Sección: C3

UNIVERSIDAD DE SANTIAGO DE CHILE

Contenido

UNIVERSIDAD DE SANTIAGO DE CHILE

Introducción

El paradigma a estudiar es el orientado a objetos, en donde para comprender cabalidad fue asignado un proyecto de laboratorio, que cual consiste en la simulación de una plataforma estilo “Photoshop”. El presente informe tiene como objetivo comprender en su totalidad el código resultante de la implementación, para ello el informe dispondrá del análisis del problema, el diseño de la solución, las consideraciones de implementación, instrucciones de uso, los resultados, la autoevaluación y finalmente la conclusión.

Descripción del problema

Los softwares de edición de fotografías (similares a Photoshop) permiten editar o manipular fotografías, para fines prácticos y profundizar los conocimientos adquiridos del paradigma funcional, el proyecto solo se concentrará en editar fotografías del tipo RGB-D, las cuales son fotografías con color, pero que adicionalmente contiene información de la profundidad en un tamaño de 3 dimensiones.

Descripción y uso del paradigma

El Paradigma Orientado a Objetos (POO) se caracteriza por ser un paradigma basado en el concepto de clases y objetos.

Dentro del POO existen las clases, las cuales se componen de atributos, los cuales son características de un objeto, también tienen métodos los cuales son comportamientos. Un objeto es una instancia de una clase, de esta forma, se puede crear objetos, como por ejemplo una persona, esta posee atributos como lo son: El nombre, el color de piel, la estatura, etc. Además, tiene comportamientos como lo son: Hablar, comer, dormir, caminar, etc., esto ayuda a tener abstracciones de la realidad a un código de manera más precisa. Adicionalmente, se debe destacar que el análisis y diseño POO de una solución informática deben resolver dos preguntas la cuales son: El ¿Qué? Y el ¿Cómo?, por tal motivo, este paradigma puede trabajar en conjunto con otros, como lo pueden ser el paradigma imperativo, el orientado a eventos, etc.

Algunos aspectos de la POO que fueron utilizados son:

Composión: Es una fuerte relación entre dos objetos, en la cual una depende de la otra, una forma de poder identificarla es la frase “parte de”, por ejemplo: un motor es parte de una moto, dejando en claro que la moto depende del motor para existir, sin embargo, el motor puede existir sin necesitar de otras clases.

Herencia: Se puede definir como una relación entre clases, donde la superclase hereda atributos y métodos a subclases, esto se utiliza para no repetir no código innecesariamente, también así disminuye la posibilidad de cometer errores entre cada código reescrito, también simplifica la forma de comprender un problema, también porque un cambio en la superclase afecta a todas las clases hijas.

Clase Abstracta: Es una clase que no se puede instanciar, y también, solo puede ser utilizada por las subclases.

Interface: Es una clase que no puede ser instanciada, no posee atributos y solo poseen la declaración de los métodos que la implementa.

Análisis del problema

La presente plataforma de edición de imágenes funciona con una imagen, la cuál esta compuesta por pixeles de tipo bitmap, hexmap y/o pixmap, de esta forma, el problema se puede dividir en 4 entidades fundamentales (TDA) sobre los cuales se desarrollará todo el proyecto y sobre estos operarán todas las funciones que puede hacer un editor de fotos.

Datos de entrada:

El usuario deberá ingresar los valores de una imagen, con los respectivos atributos de cada tipo de pixel, además la imagen debe ser entregada en orden, con sus respectivas coordenadas, de lo contrario la imagen podría no quedar como se desea.

Las imágenes poseen una estructura especifica, la cual está compuesta por un ancho, un alto, el/los color (es) que tendrá dicho pixel.

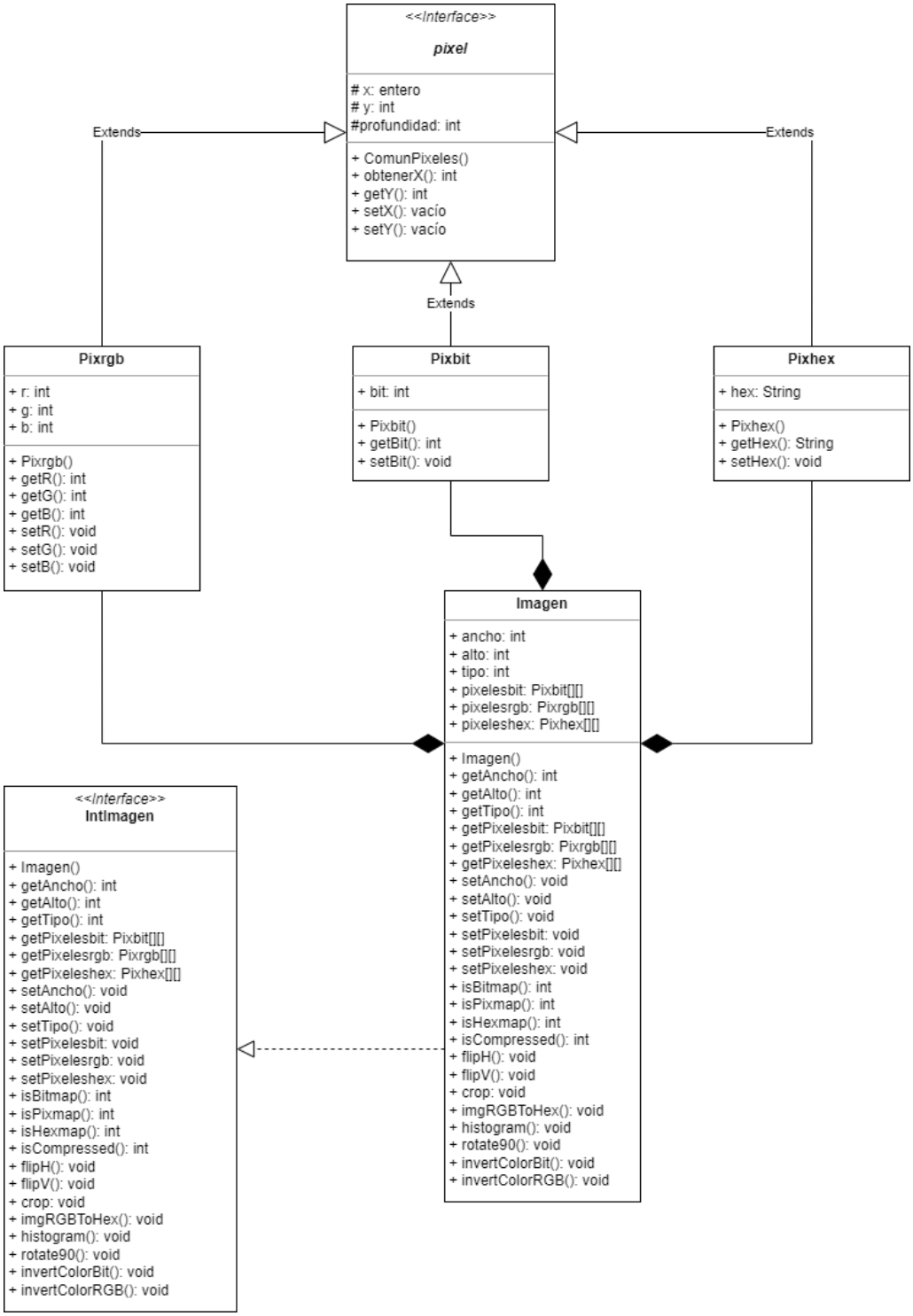
Una vez creada la imagen, sobre esta se puede hacer distintas operaciones, las cuales son:

- Recortar imagen.
- Invertir la imagen horizontalmente.
- Invertir la imagen verticalmente.
- Comprimir imagen en base a la eliminación del color con mayor frecuencia.
- Convertir a hexadecimal (Solo si es de tipo Pixmap).
- Visualizar la imagen.
- Rotar la imagen 90° a la derecha.
- Histograma.
- Descomprimir la imagen en base a la restitución del color con mayor frecuencia.
- Convertir imagen a blanco y negro.
- Convertir imagen a escala de grises.
- Editar una imagen a partir de la aplicación de funciones especiales sobre pixeles.
- Separar capas de una imagen 3D en base a la profundidad. De esta forma desde una imagen 3D se puede devolver a una lista de imágenes 2D.
- Redimensionar imágenes

Datos de salida:

Una vez el usuario haya ingresado la imagen pixel por pixel de manera correcta, podrá comenzar a realizar operaciones. Se debe destacar que cada función modificara la imagen ingresada y también solo se puede trabajar con una imagen a la vez, pero se puede volver a ingresar una nueva imagen si se desea, solo que se perderá la imagen anterior.

Diseño de la solución:



En el diagrama UML se puede ver la relación entre herencia que existe entre cada tipo de píxel y el "pixel genérico" que es el que contiene las coordenadas y la profundidad de cada píxel. En el diagrama también se puede ver a relación de composición entre los tipos de pixeles y la imagen, dado que sin pixeles la imagen no podría existir, esta clase a su vez presenta una interfaz, la cual contiene todos los métodos de la imagen.

Para realizar operaciones con imágenes, debe haberse creado una previamente, en caso de que el usuario desee probar la edición de fotografías puede hacer uso de un ejemplo, este es una imagen de 2 x 2 compuesta por 4 pixeles de tipo pixrgb-d (pixmap), donde cada píxel tiene sus colores, profundidad y coordenadas correspondientes.

Cada imagen posee la siguiente estructura: (ancho alto tipo [pixbit-d* | pixrgb-d* | pixhex-d*]), cabe destacar que la imagen puede recibir cualquier tipo de pixel, sin embargo, los que sean ingresados deberán ser homogéneos, es decir, todos deben ser del mismo tipo, de lo contrario el programa dará aviso y se deberán ingresar los valores nuevamente.

Los pixeles estarán compuestos de la siguiente forma:

- Pixbit = (posiciónX, posiciónY, bit, profundidad,), donde bit es un valor 0 o 1.
- Pixrgb = (posiciónX, posiciónY, red, green, blue, profundidad) , donde red, green y blue son valores entre 0 y 255.
- Pixhex = (posiciónX, posiciónY, hex, profundidad) , donde hex es un numero rgb en formato hexadecimal.

Cada uno de los pixeles será tratado como un TDA (clase) diferente, el TDA image (clase imagen) hará uso de las funciones presentes es los TDAs pixeles (clase pixhex, pixrgb, pixbit).

Sobre las imágenes se podrán llevar a cabo funciones de tipo edición, agregación, eliminación, entre otras. Las previamente señaladas son:

Funciones de consulta

1. bitmap? : Permite determinar si una imagen es de tipo bitmap-d.
2. pixmap? : Permite determinar si una imagen es de tipo pixmap-d.
3. hexmap? : Permite determinar si una imagen es de tipo hexmap-d.
4. compressed? : Permite determinar si una imagen ha sido comprimida.
5. histogram : Permite obtener un histograma de frecuencias a partir de los colores de cada imagen.

Funciones de edición

1. image : Crea una imagen a partir del ancho, alto y los pixeles
2. flipH : Voltea una imagen horizontalmente.
3. flipV : Voltea una imagen verticalmente.
4. crop : Permite recortar una foto a partir de un cuadrante.
5. imgRGB->imgHex : transforma una imagen desde una representación RGB a una representación HEX.

6. rotate90 : Rota una imagen 90° a la derecha.
7. compress : Comprime una imagen eliminando aquellos pixeles con el color más frecuente.
8. invertColorBit : Permite obtener el valor del bit opuesto.
9. invertColorRGB : Permite obtener el color simétricamente opuesto en cada canal dentro de un pixel.
10. changePixel : Permite reemplazar un pixel en una imagen.
11. image->string : Transforma una imagen a una representación string.
12. depthLayers : Permite separar una imagen en capas en base a la profundidad en que se sitúan los pixeles.
13. decompress : Permite descomprimir una imagen comprimida.

Aspectos de implementación

El lenguaje de programación a usar el Java, a través del IDE Netbeans (versión 12.0), la versión de JDK utilizada es la 11.0.16. Los motivos de la elección de este IDE es por la comodidad que este ofrece (criterio personal). La versión del grade utilizada es la 6.3. También se debe mencionar que no se utilizaron bibliotecas externas.

La estructura del proyecto es 7 archivos diferentes, 5 para cada TDA y 1 para la interfaz y una clase abstracta para los pixeles:

- TDA image : Encargado de crear las imágenes.
- TDA bitmap : Encargado de crear pixeles de tipo pixbit-d.
- TDA pixmap : Encargado de crear pixeles de tipo pixrgb-d.
- TDA hexmap : Encargado de crear pixeles de tipo pixhex-d.
- Main : Encargado de ejecutar pruebas de funciones.
- InterImagen : Interfaz de la imagen.

Las clases mencionadas se pueden ver en el diagrama UML previo.

Ejemplos de uso

La imagen a probar es la siguiente:

```
******************Imagen creada*****************
Ancho:3  Alto:3
[1,1,1] [2,2,2] [3,3,3]
[4,4,4] [5,5,5] [6,6,6]
[7,7,7] [8,8,8] [9,9,9]
```

- isBitmap:

```
No es del tipo bitmap
```

- isPixmap:

```
Es del tipo pixmap
```

- isHexmap:

```
No es del tipo bitmap
```

- isCompressed:

```
La imagen no esta comprimida
```

- flipH:

```
**Imagen volteada Horizontalmente**
Ancho:3, Alto:3

[3,3,3] [2,2,2] [1,1,1]
[6,6,6] [5,5,5] [4,4,4]
[9,9,9] [8,8,8] [7,7,7]
```

- flipV:

```
**Imagen volteada Verticalmente**
Ancho:3, Alto:3

[7,7,7] [8,8,8] [9,9,9]
[4,4,4] [5,5,5] [6,6,6]
[1,1,1] [2,2,2] [3,3,3]
```

- Crop(coordenadas 0,1,1,2)(la imagen fue rotada 90 grandos antes de ser cortada):

```
**Imagen recortada**
Ancho:2, Alto:2

[4,4,4] [1,1,1]
[5,5,5] [2,2,2]
```

- Rotate90:

```
**Imagen rotada 90 grados a la derecha**
Ancho:3, Alto:3

[7,7,7] [4,4,4] [1,1,1]
[8,8,8] [5,5,5] [2,2,2]
[9,9,9] [6,6,6] [3,3,3]
```

-

Posibles errores

Al momento de crear una imagen los pixeles se deben colocar en orden, ya que, de lo contrario, el programa podría entregar valores de manera errónea.

Autoevaluación

Probando el código se pueden ver algunas funciones, algunas de ellas no funcionan al

100%, sin embargo, otras funcionan perfectamente. En la escala de la autoevaluación:

- 0: No realizado
- 0.25: Implementado con problemas mayores.
- 0.50: Implementado con funcionamiento irregular.
- 0.75: Implementado con problemas menores.
- 1: Implementación sin problemas.

Requerimientos Funcionales	Evaluación
Clases y estructuras	1
Menú	1
image	1
isBitmap	1
isPixmap	1
isHexmap	1
isCompressed	1
flipH	1
flipV	1
crop	1
imgRGBToHex	1
histogram	0.25
rotate90	1
compress	0
changePixel	1
invertColorBit	1
invertColorRGB	1
image->string	0
depthLayers	0
decompress	0

UNIVERSIDAD
DE SANTIAGO
DE CHILE

Conclusión

Se puede concluir que se pudieron llevar a cabo la mayoría de las funciones, además, la esencia de la Programación Orientada a Objetos se pudo comprender, para esto, los diagramas de clase fueron muy útiles, también se pudo comprender la relación que existe entre el código, el enunciado del problema y el diagrama UML. Durante el curso de paradigmas de programación, se estudiaron 3 paradigmas, el paradigma funcional, el lógico y el orientado a objetos, dentro de los cuales se pudo abordar la misma problemática, por ende, se pueden ver algunos puntos fuertes de cada uno, en el paradigma funcional se podían detectar errores de manera mucho más rápida debido a la inexistencia de variables, además, al estas no existir y tener que ingresar funciones como parámetros se espera que el valor que esta retorne sea inmutable, también el hecho de que sea más enfocado más al ¿qué hacer? En vez del ¿cómo hacerlo? Facilita mucho el trabajo, en el paradigma lógico, lo que más se puede destacar la lógica matemática, que se basa en predicados y las relación entre ellos, y finalmente el paradigma orientado a objetos, el cual permite el uso de clases y objetos, los cuales entregan una mejor abstracción de los objetos del mundo real a un programa, además este soporta paradigmas que la complementen, un claro ejemplo de esto es el paradigma imperativo.

UNIVERSIDAD DE SANTIAGO DE CHILE

Referencias

Gonzales, R. (2022). Proyecto Semestral de Laboratorio. Recuperado el 2 de diciembre de 2022, de https://docs.google.com/document/d/1ymOs4hi2NYhFbJDJIkgwZkM-qBd8MF5_BkwTk9d7qcg/edit?pli=1