UNIVERSIDAD
DE SANTIAGO
DE CHILE

# Simulación de una plataforma

# estilo “Photoshop”

## Paradigma Funcional

Profesor: Miguel Truffa

Estudiante: Nicolás
Gajardo Ponce

Sección: C3

UNIVERSIDAD DE SANTIAGO DE CHILE

# Contenido

UNIVERSIDAD DE SANTIAGO DE CHILE

# Introducción

El primer paradigma a estudiar es el funcional, 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 funcional se caracteriza por estar más orientado al ¿qué hacer? antes que al ¿cómo hacerlo?, además se puede destacar la ausencia de variables, por ende, se espera que las funciones retornen un mismo valor, es decir inmutable.

Dentro del paradigma funcional todo es reutilizable, por tal motivo, las funciones de los TDA fueron creadas para ser reutilizadas posteriormente, hasta finalizar el proyecto, un claro ejemplo en este proyecto es la función n°2 que se debe usar hasta la función n°20, es decir, como base en todo el proyecto. Otro aspecto que se puede destacar es la trazabilidad de los errores, debido a que, al ser todo una función, se puede identificar la función que está dando problemas, en vez que de modificar una variable y al final no saber no fue que cambió todo y apareció un dato erróneo.

# 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 entregar una imagen en el orden correspondiente, de lo contario, el

programa podría no funcionar correctamente formando imágenes incorrectas y realizar todas las funciones mal.

Las imágenes poseen una estructura especifica, la cuál 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.
- Visualizar la imagen.
- Rotar la imagen 90° a la derecha.
- Histograma.
- Descomprimir la imagen en base a la restitución del color con mayor frecuencia.
- Aplicar operaciones como las anteriores a un área seleccionada dentro de la imagen.
- 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 correctamente podrá comenzar a realizar operaciones. Se debe destacar que solamente se puede hacer una operación a la vez. Cabe destacar que la gran mayoría de funciones retornan la imagen editada, es decir, solo los pixeles de esta

Diseño de la solución:

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 casa pixel tiene sus colores, profundidad y coordenadas correspondientes.

Cada imagen posee la siguiente estructura: (ancho alto [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.

Los pixeles estarán compuestos de esta forma:

- Bitmap = (pixbit-d ancho alto [0|1] profundidad)
- Pixmap = (pixrgb-d ancho alto red green blue profundidad) , donde red, green y blue son valores entre 0 y 255.
- Hexmap = (pixhex-d ancho alto hex profundidad) , donde hex es un numero rgb en hexadecimal.

Cada uno de los pixeles será tratado como un TDA diferente, por ende cada uno posee funciones propias de su TDA, sin embargo, el TDA image hará uso de las funciones presentes es los TDAs pixeles.

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 : Invierte una imagen horizontamente.
3. flipV : Invierte 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. edit : Permite aplicar funciones especiales a las imágenes.

9. invertColorBit : Permite obtener el valor del bit opuesto.
10. invertColorRGB : Permite obtener el color simétricamente opuesto en cada canal dentro de un pixel.
11. adjustChannel : Permite ajustar cualquier canal de una imagen con pixeles pixrgb-d, icluido el canal de profundidad d.
12. image->string : Transforma una imagen a una representación string.
13. depthLayers : Permite separar una imagen en capas en base a la profundidad en que se sitúan los pixeles.
14. decompress : Permite descomprimir una imagen comprimida.

## Aspectos de implementación

El lenguaje de programación a usar es Scheme a través del interprete Dr. Racket. Los motivos de la elección del interprete es por la simplicidad y practicidad que este posee para trabajar con este idioma y paradigma. Además, cabe mencionar que el proyecto fue creado bajo la versión 8.6 del interprete. También se debe mencionar que no se usaron bibliotecas en particular

La estructura del proyecto es 5 archivos diferentes, uno para cada TDA:

- 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.

## Ejemplos de uso

Img es una imagen creada con la función image, los parámetros que se usaron en su construcción fueron:

```
(define img1 (image 2 2
                    (pixrgb-d 0 0 255 0 0 10))
                    (pixrgb-d 0 1 0 255 0 20))
                    (pixrgb-d 1 0 0 0 255 10))
                    (pixrgb-d 1 1 255 255 255  1))
 ))
```

- (bitmap? img1)  Retorna #f

- (pixmap? img1) Retorna #t
- (hexmap? img1) Retorna #f
- (compressed? img1) Retorna #f
- (flipH img1) Retorna '((0 0 0 255 0 20) (0 1 0 255 0 10) (1 0 255 255 255 1) (1 1 0 0 255 10))

### Posibles errores

Al momento de crear una imagen los pixeles se deben colocar en orden, ya que de lo contrario, el programa entregará coordenadas distintas a las esperadas.

## 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 |
|---|---|
| TDAs | 1 |
| image | 1 |
| bitmap? | 1 |
| pixmap? | 1 |
| hexmap? | 0.5 |
| compressed? | 1 |
| flipH | 1 |
| flipV | 0.25 |
| crop | 0 |
| imgRGB->imgHex | 0 |
| histogram | 0 |
| rotate90 | 0 |
| compress | 0 |
| edit | 0 |
| invertColorBit | 0 |
| invertColorRGB | 0 |
| adjustChannel | 0 |
| image->string | 0 |
| depthLayers | 0 |
| decompress | 0 |

UNIVERSIDAD
DE SANTIAGO
DE CHILE

# Conclusión

Se puede concluir que no se pudieron implementar todas las funciones, sin embargo, la esencia de la programación funcional si se pudo comprender. Se espera que los próximos proyectos de laboratorio se puedan llevar a cabo completamente, para así poder comparar los tipos de paradigmas a estudiar durante el curso y saber bien cual resulta más útil para casa situación.

UNIVERSIDAD DE SANTIAGO DE CHILE

# Referencias

*Gonzales, R*. (2022). Proyecto Semestral de Laboratorio. Recuperado el 26 de septiembre de 2022, de https://docs.google.com/document/d/1hUAooKwBj3TWv-yuzBZtNbuaC8iNkzOZdbLpD8P9B8c/edit