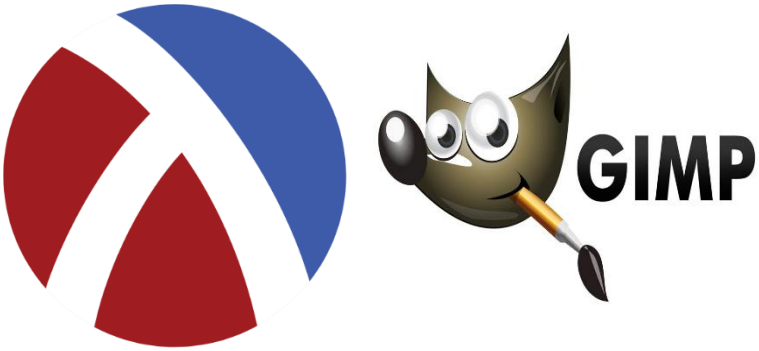
Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería en Informática
Paradigmas de Programación

INFORME DE LABORATORIO 1:

GIMP MEDIANTE SHEME



- **Autor:** Gabriel Ojeda.
- **Profesor:** Gonzalo Martínez.

- **Fecha:** 26 de septiembre del 2022.

Contenido

1. INTRODUCCIÓN

En el presente informe se darán a conocer los elementos para poder adentrarse al problema que se presenta, el cual consiste en replicar las famosas aplicaciones de edición de imágenes y videos GIMP o Adobe Photoshop en lenguaje shceme basándose en la particularidad de este, el paradigma funcional, de esta manera se mostraracómo se abordó el tema, el enfoque que se le dio y como se llegó a una solución que permitió realizar un programa con la parcialidad de los requisitos básicos completados en su totalidad, además se daráuna conclusión donde se hará una autocrítica y como mejorar para el próximo laboratorio.

DESCRIPCIÓN DEL PROBLEMA

Se pide desarrollar una simulación de un software de edición o manipulación de imágenes digitales, el cual le permite a un usuario realizar distintas operaciones sobre éstas. Tal y como son los softwares GIMP y Adobe Photoshop. Existen distintos tipos de operaciones tales como rotar una imagen, invertirla, rotarla, retocarla, transformarla y redimensionarla, entre otras.

El proyecto se concentrará en trabajar en imágenes RGBD o RGB-D, esto es, imágenes que además de tener información en el espacio de colores (R)ed, (G)reen, (B)lue, contiene información de la profundidad (D)epth en un espacio tridimensional. Al incorporar la dimensión D (profundidad) capturada a través de una cámara especializada, sería posible saber más sobre los detalles del rostro, proyección de la nariz, sombrero, distancia del espejo en la parte posterior, etc. Incluso sería posible construir una representación tridimensional del rostro.

Otro elemento a considerar es que esta versión del software operará con representaciones sencillas de imágenes, En particular, se consideran bitmaps-d (para imágenes donde cada pixel o pixbit puede tomar el valor 0 o 1), pixmap-d (para imágenes donde cada pixel o pixrgb es una combinación de los valores para los canales R, G y B) y hexmaps-d (donde cada pixel o pixhex expresa la información del color del pixel a través de un valor único hexadecimal de 6 valores)

Se debe implementar el software en el paradigma funcional, específicamente en el lenguaje de programación Scheme y el compilador Dr. Racket.

DESCRIPCIÓN DEL PARADIGMA

La programación funcional, como su nombre lo dice, trabaja en términos de funciones, la cual es la unidad de abstracción y programación de este paradigma.
Una función se define como una transformación entre elementos de un dominio y un recorrido, el dominio es un conjunto de elementos que serán transformados o vinculados a un conjunto recorrido.

Este paradigma no trabaja con variables aunque se pueden declarar dentro de funciones o sobre declarar funciones ya existentes, lo cual no es éticamente correcto dado que el fundamento principal de este paradigma es realizar funciones sobre funciones por ende solamente trabaja con funciones y para este trabajo se considera el uso del lenguaje de programación funcional Scheme, cabe destacar que Scheme no es un lenguaje puro de programación funcional, ya que tiene aspectos de la programación imperativa. Este paradigma está situado en la programación declarativa, que se basa en el "¿Qué?", en cambio la imperativa se centra en "¿Cómo llegamos al qué?" Esto quiere decir que el paradigma funcional se enfoca en el resultado y no en el procedimiento, lo que implica un mayor nivel de abstracción. Un ejemplo puede ser el siguiente "Con este paradigma las funciones serán tratadas como ciudadanos de primera clase. Las funciones podrán ser asignadas a variables además podrán ser utilizadas como entrada y salida de otras funciones." (Pérez, 2019).
Este paradigma trabaja principalmente con:

- **Currificación:** Es la evaluación de una función de "n" argumentos, transformada a la evaluación de una secuencia de funciones de un argumento. Para trabajarlo, se ocupan funciones anónimas bajo el concepto de funciones de orden superior, ya que se van estableciendo funciones de un argumento que retornan otra función de un argumento, así sucesivamente hasta pasar todos los parámetros.
- **Recursividad:** En este paradigma es fundamental, el cual es un método de resolución de problemas, que se basa en que la solución de un problema depende de soluciones más pequeñas del mismo problema. Una función recursiva es la que se expresa en términos de sí misma, específicamente en este trabajo se ocupa solamente la recursión natural y de cola.
- **Composición de Funciones:** Es aquella operación a través de la cual dos funciones generan una tercera función.
- **Funciones Anónimas:** Las funciones anónimas y el paradigma funcional en sí, trabajan bajo el Cálculo Lambda, estas funciones se expresan sin un nombre, bajo una variable (no en el sentido del paradigma imperativo) y una transformación de esta que aplica la función.
- **Funciones de Orden Superior:** Son aquellas que operan sobre funciones, esto quiere decir que una función puede tener por dominio otra función, como también puede devolver unafunción, en este contexto, se tiene como ejemplo la composición de funciones.

OBJETIVOS

El objetivo del proyecto se basa en aprender y utilizar los conceptos del paradigma funcional, como también el uso del lenguaje de programación Scheme. Con esto, al terminar el proyecto, tener un buen desempeño con el paradigma, para poder ocuparlo en futuros aspectos laborales.

Otro objetivo es el desarrollo del software en el lenguaje Scheme, que tiene como propósito llevar a cabo los conocimientos del paradigma funcional y su correcta utilización bajo el lenguaje de programación Scheme, para lograr el objetivo principal.

Finalmente comprender en profundidad la lógica de programación y construir un software similar a GIMP.

2. DESARROLLO

ANÁLISIS DEL PROBLEMA

Sabemos que hay que desarrollar un software que simule la aplicación de edición de imágenes GIMP y desarrollar algunas de las funciones principales de este, como recibir imágenes en diferentes formatos, los cuales están especificados en pixbit-d, pixhex-d y pixrgb-d, rotar dichas imágenes, trasladar, comprimir, expandir, y modificaciones en general.

Para esto se requiere generar funciones o estructuras que puedan realizar todas estas labores.

El objeto a estudiar en primera instancia es una imagen, la cual puede tener 3 diferentes formatos y todos deben ser soportados por el sistema. Teniendo en cuenta esto podemos decir que:

Una imagen puede ser solo de un tipo de pixels, sin embargo, los 3 tipos de pixels tienen valores comunes, como las coordenadas x e y, y la profundidad d, por lo tanto, para hacer mas optimo el programa vamos a tener selectores y modificares transversales para estos elementos y otros particulares para cada tipo de pixel.

Por otro lado, como lo mencionan en el enunciado una imagen es tridimensional, tiene un ancho, un largo y además una lista que contiene los datos del tipo de pixel, por lo que cada pixel va a ser un TDA aparte pero que en su conjunto pueden formar el TDA imagen y así abarcar todos los tipos.

Para las funciones de pertenencia y modificación para una imagen, pueden variar dependiendo del tipo de pixel, por lo que una sola función principal, tendría que ser capaz de validar una acción por igual para los 3 tipos de imágenes, por ende, para cada función principal de la imagen se requiere de funciones particulares que se ejecutaran dependiendo de cada condicional del pixels.

DISEÑO DE LA SOLUCIÓN

En primera instancia tendremos 4 archivos TDAs los cuales serán:

TDA_image: que contiene selectores, modificadores y funciones de pertenencia, pero solo de imágenes.

Los TDAs pixels, ya sea bit, hex, o rgb, que van a tener también selectores, modificadores y funciones de pertenencia, pero asociados a su respectivo pixel.

Luego tendremos un archivo de FuncionesBase, en donde se almacenarán selectores, modificadores que son transversales para todos los TDAs, como obtener la coordenada x, la coordenada y o la profundidad d de los distintos pixeles, como también, intercalar valores de las coordenadas x o y independiente el tipo de imagen, entre otras.

Finalmente, el archivo main, va a contener el llamado de todas las funciones solicitas en el enunciado, por ende, este archivo debe exportar las funciones de todos los demás archivos ya que de aquí se obtendrán los retornos de las funciones principales.

ASPECTOS DE IMPLEMENTACIÓN

COMPILADOR

Para aspectos de este programa, se requiere usar el lenguaje Scheme y se decide usar el compilador Dr. Racket, ya que es una herramienta bastante útil al mostrar errores y permite hacer un debug, lo que resulta muy útil a la hora de buscar errores en el código. La biblioteca que se debe utilizar es sólo la de Scheme, no se permite utilizar funciones de Racket que no pertenezcan a Scheme. Para efectos de este laboratorio la implementación se basa en el trabajo de listas.

ESTRUCTURA DE CODIGO

Para estructurar el código en archivos independientes, se usan las siguientes funciones para importar y exportar en Dr. Racket, estas son “require” y “provide”. La función require importa las funciones exportadas desde otro archivo “.rkt”. La función provide exporta una función específica o todas las funciones del código fuente. Para este programa, se estructuran 7 archivos: “main.rkt”, "FuncionesBase.rkt", "TDA_image.rkt", "TDA_pixbit.rkt", " TDA_pixhex.rkt", " TDA_pixrgb.rkt" y el “prueba.rkt” (el nombre del archivo omite la estructura del rut_apellidos para este caso). Los archivos TDAs contiene las funciones principales de cada tipo de pixel, mientras que el archivo FuncionesBase contiene modificadores que son transversales y finalmente el main tiene los modificadores que realizan llamados a todos los otros archivos.

INSTRUCCIONES DE USO

EJEMPLO DE USO

Para ocupar el programa, se deben agregar las llamadas a las funciones en el archivo “main.rkt”, estando en el archivo se debe compilar. Luego de compilar se deben crear las imágenes en el formato establecido en el enunciado dependiendo del tipo de imagen que desee, ya sea pixbit, pixhex o pixrgb. Por ejemplo para pixbit tendríamos el siguiente Código.

```
“(define img1 (image 2 2
    (pixbit-d 0 0 0 10)
    (pixbit-d 0 1 1 20)
    (pixbit-d 1 0 1 10)
    (pixbit-d 1 1 0 255)))”
```

Luego de crear la imagen ya puede interactuar con las demás funciones realizando el llamado de estas como, por ejemplo:

```
“(bitmap? img1)”
“(flipH img1)”
 etc…
```

RESULTADO ESPERADO

Se espera que el programa sea 100% funcional siguiendo la estructura asignada, dejando una estructura acabada de los TDA que se deben implementar, con el formato visto anteriormente. Tras cumplir con las estructuras, se espera que se haga un correcto trabajo con listas y recursión al crear los comandos, debido a que es lo fundamental para implementarlos.

POSIBLES ERRORES

Los posibles errores que se pueden producir en el programa son los siguientes:

- Si el usuario inicializa la imagen sin respetar el formato asignado, el programa no funciona.
- Si el usuario no respeta el formato asignado a cada parámetro de entrada en los comandos, el programa no funciona.

Al probar el programa, solamente se encuentra ese posible error, por lo demás ha funcionado correctamente con todas las funcionalidades que se implementaron.

RESULTADOS Y AUTOEVALUACIÓN

RESULTADOS OBTENIDOS

Los resultados obtenidos, fueron los esperados, ya que se implementó de buena forma la simulación, excepto por el hecho de que no se alcanzaron a crear la totalidad de las funciones solicitadas.

Al probar el programa con todos los ejemplos que se pueden hacer, sólo dejó de funcionar en lo que se menciona anteriormente, sin contar ese punto y respecto a las pruebas que se hicieron, se puede decir que el programa es completamente funcional y respeta el paradigma solicitado.

AUTOEVALUACIÓN

Se evalúan las funcionalidades, de la siguiente forma:

0: No realizado – 0.25: Funciona 25% de las veces – 0.5: Funciona 50% de las veces – 0.75: Funciona 75% de las veces – 1: Funciona 100% de las veces.

FUNCIONALIDADES	EVALUACIÓN
TDA PIXBIT	1
TDA PIXHEX	1
TDAPIXRGB	1
TDA IMAGE	1
MAIN	0.5

En el archivo main todas las funciones existentes funcionan al 100% sin embargo no esta el totalidad de las funciones solicitadas en el proyecto, es por esta razón que la asignación es de 0.5. sin embargo, en los demás TDAs, existen todas los selectores, modificadores y pertenencias asociadas a cada uno y funcionando en su totalidad.

3. CONCLUSIÓN

Se puede concluir que el resultado de este primer laboratorio fue el esperado ya que se pudo realizar todas las funciones básicas para cada TDA , como los son los constructores, selectores, modificadores y pertenencia. Sin embargo, se requiere reforzar más en alcanzar a hacer las funciones que son de manera libre para alcanzar la nota máxima.

Los problemas que se presentaron fueron que en un principio se tenía una idea de cómo realizar este laboratorio y esta fue cambiando demasiado con el paso del tiempo, hasta llegar a la última idea que fue la que se dejó (Luego de reestructurar varias veces el código y los archivos) y gracias a esta se pudo realizar al 100% el laboratorio (en relación con lo funcional), pero no el 100% de las funcionalidades solicitadas.

Luego de ordenar todas las ideas y empezar a programar, las complicaciones fueron pocas, errores que se pasaban por detalles y reparaciones mínimas en implementaciones de cada función, cabe destacar que hubo muchos errores, pero ninguno que haya tomado un tiempo significativo a la hora de solucionarlo.

4. REFERENCIAS

Gonzalo Martínez, Víctor Flores y Roberto González. (2022). Programación funcional, Paradigmas de programación. Sitio web: CAMPUS VIRTUAL (usach.cl)