UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMATICA

INFORME LABORATORIO 1:

Editor de imagen en Scheme

Nombre: Jean Paul Rojas Ramos de Rosas

Fecha: 26-09-2022

Profesor: Miguel Truffa

Índice

1. Introducción:

Siempre hay problemáticas o necesidades en el mundo y siempre se suele necesitar ayuda de la programación para contrarrestarlas o hacerlas mas manejables o simplemente implementar una nueva forma de solucionarlas. Si bien esto no es mas que practica y hay mas de una forma para enfrentar las diferentes problemáticas y encontrarles una solución concreta, o que con el tiempo puede ir mejorando y evolucionando, esto es conocido como paradigma de programación. En esta ocasión el paradigma para enfrentar esta problemática es conocido como paradigma funcional, aplicado en el lenguaje Scheme, buscando una óptima forma de emplear el trabajo solicitado.

2. Descripción del problema:

Se solicita implementar un Photoshop en Scheme, con funciones que se le aplican a imágenes que mayormente se basan en iterar pixeles y su contenido, transformaciones de imágenes de un tipo a otro, para esto existen distintos tipos de formatos para los pixeles, en conjunto a una imagen que contiene estos con sus respectivas dimensiones como representación se tienen los siguientes tipos:

Image: Una imagen la cual contiene un ancho, un alto y sus pixeles que son homogéneos, esto quiere decir que todos deben ser del mismo tipo, a esta imagen se le aplicaran funciones implementadas que cambiarían sus características por completo.

Pixbit-d: Un tipo de píxel, con 4 parámetros los 2 primeros son sus posiciones x e y que ocuparan en la imagen, el tercero un bit por lo tanto solo puede contener un 0 o 1, y como ultimo parámetro la profundidad.

Pixrgb-d: Un tipo de píxel, con 6 parámetros los cuales los 2 primeros son posiciones x e y dentro de la imagen, los siguientes 3 datos son valores rgb que varían entre 0 y 255, como ultimo parámetro tenemos la profundidad.

Pixhex-d: Un tipo de píxel, con 4 parámetros los cuales los 2 primeros son sus posiciones x e y en la imagen, el tercer parámetro es una representación hexadecimal de los colores, como ultimo parámetros tenemos su profundidad.

Funciones: Funciones las cuales se pide implementar para poder aplicarle a las imágenes creadas y que estas puedan trabajar con la imagen.

El programa debe ser implementado en Scheme utilizando el paradigma funcional.

3. Descripción del Paradigma:

Si hablamos de paradigma funcional, hablamos de funciones, todo lo que aplique en el paradigma funcional esta en funciones, sin existencias de variables que almacenen datos, solo encapsulamientos de funciones que devuelven dependiendo de las entradas distintas salidas.

Hablando de las entradas y salidas estas son conocidas como dominio y recorrido donde el dominio es todo lo que entra a las funciones mientras que el recorrido es que se obtiene con este dominio llegando así a un tipo de salida dependiendo de lo que haga cada función obteniendo así el resultado con los cambios que se le quieran aplicar.

Este paradigma basa su estructura en código con ciertos tipos de características propias, tales como funciones anónimas las cuales no tienen nombre pero se ocupan para ciertas tipos de variaciones de los parámetros entregados en estas, en Scheme no existen los ciclos por lo que queda ocupar recursión para poder llevar a cabo ciertas funciones de recorrido y obtención de datos de ciertas listas, la currificación por otra parte se aplica demasiado para esta implementación puesto que hay muchas funciones que solo necesitan un parámetro y por dentro de esta función se obtienen datos de este parámetro ingresado, así componiendo funciones dentro de otras se pueden variar los datos obtenidos de otra función encadenando así funciones dentro de funciones para obtener lo requerido.

4. Análisis del problema:

La creación de una imagen no es del todo fácil, hay muchas restricciones para llevar a cabo en este caso, en el caso de que una imagen sea un bitmap asegurarse que sus bit solo se basen entre 0's y 1's, en el caso de que sea pixmap asegurarse que sus colores r g b tengan valores entre 0 y 255 para poder simular un color y en el caso de ser hexmap este tenga su dato de color expresada de forma hexadecimal para poder interpretarlo y darle color con los datos que se crearon, por lo mismo las restricciones para crear una imagen de cierto tipo tiene que tener ciertas condiciones para que sea posible crearla, en el caso de crear una imagen comprimida esta podría crearse pero no se podría aplicar las funciones que se le podrían aplicar a imágenes descomprimidas puesto que la falta de pixeles puede tener unos factores en contra.

Esto podría llevar a la falla de las funciones sin antes descomprimirlas puesto que la mayoría suele estructurar sus imágenes de una manera descomprimida para poder aplicarle cambios con las funciones creadas.

5. Diseño de la solución:

La forma en la que fue creada la solución fue respetando los datos planteados, para crear una imagen debo comprobar que ambos parámetros ingresado como ancho y alto sean enteros de esta forma me aceptaría los datos, y que respetando las restricciones antes entregadas acerca de pixbit , pixrgb y pixhex se entregue una lista con estos parámetros de manera homogénea, al mismo momento de crear la imagen esta lista es ordenada automáticamente ordenando por valores de sus x e y para poder aplicar las funciones que necesitan las coordenadas ordenadas, la mayoría de las funciones creadas para este laboratorio requerían que los pixeles de la imagen estuvieran ordenados por lo tanto esto genero la idea de ordenar todos los datos para al cambiar los parámetros luego pegar una lista de coordenadas ordenadas con cada posición para que el cambio que producen las funciones para la imagen no afecte las posiciones x e y de cada imagen.

Se entregará un anexo explicando la descripción de cada funcion.

6. Aspectos de implementación:

Compilador: Dr.Racket desde la versión 6.11 o superior

Estructura código: El código se estructura con funciones para filtrar TDAimage, con selectores de TDAimage , en conjunto con las funciones para pixbit, pixhex, pixrgb y por consiguiente las funciones solicitadas a implementar.

7. Instrucciones de uso:

Para poder usar el código se deben tener todos los archivos ".rkt" en una misma carpeta puesto que usan las funciones provide y require para que las funciones sean aceptadas en otros archivos como el main.

Apretar run para ejecutar los llamados a funciones creados en el main y comprobar si los llamados a las funciones son correctos.

8. RESULTADOS:

Se logro implementar las funciones obligatorias, la mayoría funciona de manera correcta, la función crop arroja unos errores, pero es por los parámetros ingresados para cortar la imagen, se hará entrega de un anexo evaluando cada función.

La autoevaluación es 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.

Sera entregado en un anexo llamado autoevaluación.txt en la carpeta entregables

9. Conclusiones:

Terminando con el proyecto, se asume que se cumple con lo que objetivamente buscaba este laboratorio que básicamente es implementar todo en base al paradigma funcional esto fue logrado correctamente.
Logrando ampliar el pensamiento con este paradigma pensando en encapsular cada función y nuevamente evaluar el resultado, mayormente se ocupó la currificacion en muchas funciones que fue con lo que más se logro adaptar las funciones creadas.

9.1 Referencias:

1. González, A., & Página, C. (n.d.). *GUÍA BÁSICA DE SCHEME v.4*. Retrieved from http://www.gedlc.ulpgc.es/docencia/lp/documentacion/GB_Scheme.pdf
2. MIT Scheme Reference - Strings. (2022). Retrieved September 26, 2022, from Mit.edu website: https://groups.csail.mit.edu/mac/ftpdir/scheme-7.4/doc-html/scheme_7.html

Anexo:

Funciones del TDA bit

Tipo de función	Nombre	Descripción
Constructor	pixbit-d	Verifica los datos y crea una lista
Pertenencia	bitmap?	Verifica que la imagen sea un bitmap

Funciones del TDA rgb

Tipo de función	Nombre	Descripción
Constructor	pixrgb-d	Verifica los datos y crea una lista
Pertenencia	pixmap?	Verifica que sea un pixmap

Funciones del TDA hex

Tipo de función	Nombre	Descripción
Constructor	pixhex-d	Verifica los datos y crea una lista
Pertenencia	hexmap?	Comprueba que sea un Hexmap

Funciones del TDA image

Tipo de función	Nombre	Descripción
Constructor	image	Crea una imagen
Selectores	get-Ancho	Devuelve el ancho de la imagen
Selectores	get-Alto	Devuelve el alto de la imagen
Selectores	get-Pixeles	Devuelve los pixeles de la imagen
Otras	Lista-coordenadas	Devuelve una lista de coordenadas
Otras	Ord-pix-envoltorio	Devuelve los pixeles de una imagen ordenados
Funciones obligatorias	Compressed?	Revisa si una imagen esta comprimida
Otras	transformar	Devuelve una lista de pixeles sin x e y
Otras	volverEnvoltorio	Devuelve una lista lineal desde una con sublistas

Otras	juntarEnvoltorio	Devuelve una lista con los datos de una sublista dentro de otra
Otras	invertirH	Devuelve una lista agrupada dependiendo de un ancho entregado
Funciones obligatorias	flipH	Gira una imagen horizontalmente
Otras	agrupar	Devuelve una lista agrupada dependiendo de un alto entregado
Funciones obligatorias	flipV	Gira una imagen verticalmente
Otras Funciones	Filtrar-alto	Filtra una lista dependiendo de las coordenadas x1 x2
Otras Funciones	Filtrar-ancho	Filtra una lista dependiendo de las coordenadas y1 y2
Otras Funciones	anchoImpares	Devuelve el ancho de una nueva imagen
Otras Funciones	anchoPares	Devuelve el ancho de una nueva imagen
Otras Funciones	altoImpares	Devuelve el alto de una nueva imagen
Otras Funciones	altoPares	Devuelve el alto de una nueva imagen
Otras Funciones	Filtrar-puntos	Filtra una imagen en ancho y alto
Funciones obligatorias	crop	Saca un cuadrante
Otras Funciones	crearHex	Cambia los datos de unos pixeles RGB para transformarlos a hex
Otras funciones	transformarHex	Convierte de rgb a hex y los guarda en una lista
Otras funciones	Filtro-depth	Elimina el dato Depth de los pixeles
Funciones obligatorias	imgRGB->imgHex	Transforma imagen RGB a hex