Informe Laboratorio 1

Paradigma Funcional – Lenguaje Scheme

Fabián Alejandro Lizama González

Departamento de Ingeniería Informática

Facultad de Ingeniería, Universidad de Santiago de Chile

Paradigmas de Programación

Profesor Roberto González Ibañez

26 de septiembre de 2022

Índice

Introducción

En el informe se presenta el concepto de Paradigma Funcional (concepto detallado en el cuerpo del documento) aplicado al desarrollo de un software de manipulación de imágenes bajo un enfoque simplificado, bajo el lenguaje de programación “Scheme” en el intérprete “DrRacket”.

Se desarrollará este software presentando el problema, analizándolo y comentando aspectos del proceso de solución e implementación de éste junto a algunas instrucciones y ejemplos de uso para terminar con el apartado de resultados y conclusiones.

Descripción del problema

Se solicita diseñar un software de manipulación de imágenes del tipo RGBD, las cuales tienen la particularidad de adicionalmente a los canales de colores (red, green, blue) se tiene un canal de profundidad para representar imágenes en tres dimensiones, con el objetivo de poder manipular a gusto este tipo de archivos mediante distintas funciones. Para esto se requiere implementar la solución bajo el paradigma funcional e implementar una representación basada en “Tipos de Datos Abstractos” (TDAs).

Descripción del Paradigma

El Paradigma Funcional mencionado anteriormente consiste en trabajar netamente con funciones, basando la abstracción y la implementación al programar en estas mismas. Una función en simples palabras es el proceso de llevar un elemento a otro, está conformada por tres componentes esenciales, el dominio, el cual corresponde a lo que se va a transformar, el proceso, donde se trabajan los elementos de entrada del dominio, y el recorrido, que corresponde al resultado de esta transformación a él/los elementos iniciales del dominio.

Algunos aspectos de las funciones a tratar en la solución son los siguientes:

Composición de funciones: La base del paradigma funcional pues como todo son funciones, la única manera de poder construir en base a ellas es ocupando funciones dentro de otras funciones.

Recursión: Proceso donde una función se utiliza a sí repitiendo cierto proceso para realizar ciertos tipos de cálculos, en este proyecto se utiliza principalmente la recursión de cola.

Funciones anónimas: Se trata de funciones no definidas bajo un nombre especifico que se utilizan para una sola funcionalidad y no se vuelven a ocupar, debido a esto no vale la pena guardarlas en la memoria del programa.

Análisis del Problema

La complejidad del problema radica en el realizar el software solo bajo el paradigma funcional, saliendo de la zona de confort que sería la programación imperativa, bajo este principio se debe desarrollar el programa con algunas restricciones, por ejemplo, el no uso de variables, respetar el dominio y recorrido de cada función sin excepciones y cumplir con los prerrequisitos de cada función en específico.

Se implementarán abstracciones apropiadas para el problema con estructuras basadas principalmente en listas o pares, junto a sus operaciones asociadas de construcción, selección, modificación y otras.

Se busca que se logren manipular imágenes digitales bajo un enfoque simplificado, con el formato RGBD, es decir, canales de colores y además un cuarto canal de profundidad para representar imágenes en 3D, además de poder realizar las siguientes operaciones:

- Recortar imagen
- Invertir una imagen horizontalmente
- Invertir una imagen verticalmente
- Comprimir imagen en base a eliminación del color con mayor frecuencia.
- Convertir a hexadecimal
- Visualizar la imagen
- Rotar imagen en 90° a la derecha
- Rotar imagen en 90° a la izquierda
- Histograma
- Descomprimir imagen en base a 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 los pixeles
- Separar capas de una imagen 3D en base a la profundidad. De esta forma desde una imagen 3D se puede devolver una lista de imágenes 2D
- Redimensionar imagen

Diseño de la Solución

Para desarrollar el programa se utilizan cuatro TDAs:

1. TDA imagen:

   Tipo de dato abstracto que representa una imagen, está conformada por:

   ([string | int] (color) X int (w) X int (h) X list [pixbit-d | pixrgb-d | pixhex-d] (pixlist))

   - El primer elemento corresponde al color más frecuente en la imagen cuando esta se comprime en formato hexadecimal, si la imagen no está comprimida se inicializa en -1.
   - Los elementos w y h corresponden al ancho y al largo de la imagen.
   - Finalmente, el elemento pixlist corresponde a la lista de pixeles homogéneos que conforman la imagen en sí.

2. TDA pixbit-d:
   Tipo de dato abstracto que representa un píxel de formato pixbit-d, está conformado por:
   (int (pixtype) X int (x) X int (y) X int [0 | 1] (bit) X int (depth))
   - El primer elemento corresponde a una constante que indica el tipo de píxel, en este caso 0 para pixbit-d.
   - Los elementos x e y corresponden a las coordenadas del píxel en la imagen.
   - Los elementos bit y depth representan el color del píxel que puede ser blanco o negro y la profundidad que tiene el píxel en la imagen.

3. TDA pixrgb-d:
   Tipo de dato abstracto que representa un píxel de formato pixrgb-d, está conformado por:
   (int (pixtype) X int (x) X int (y) X int (r) X int (g) X int (b) X int (depth))
   - El primer elemento corresponde a una constante que indica el tipo de píxel, en este caso 1 para pixrgb-d.
   - Los elementos x e y corresponden a las coordenadas del píxel en la imagen.
   - Los elementos r g b y depth representan respectivamente el código RGB del píxel junto a su profundidad en la imagen.

4. TDA pixhex-d:
   Tipo de dato abstracto que representa un píxel de formato pixhex-d, está conformado por:
   (int (pixtype) X int (x) X int (y) X string (hexcode) X int (depth))
   - El primer elemento corresponde a una constante que indica el tipo de píxel, en este caso 1 para pixrgb-d.
   - Los elementos x e y corresponden a las coordenadas del píxel en la imagen.
   - Los elementos r g b y depth representan respectivamente el código RGB del píxel junto a su profundidad en la imagen.

Aspectos de Implementación

Para el proyecto se utiliza el lenguaje Scheme en el intérprete y editor de texto “DrRacket” el programa se organiza en un repositorio con los siguientes archivos:

image_21081166_LizamaGonzalez.rkt

Archivo de código en el lenguaje de programación Scheme con la implementación del TDA image y sus operaciones asociadas.

pixbit-d_21081166_LizamaGonzalez.rkt

Archivo de código en el lenguaje de programación Scheme con la implementación del TDA pixbit-d y sus operaciones asociadas.

pixrgb-d_21081166_LizamaGonzalez.rkt

Archivo de código en el lenguaje de programación Scheme con la implementación del TDA pixrgb-d y sus operaciones asociadas.

pixhex-d_21081166_LizamaGonzalez.rkt

Archivo de código en el lenguaje de programación Scheme con la implementación del TDA pixhex-d y sus operaciones asociadas.

pruebas_21081166_LizamaGonzalez.rkt

Archivo de código en el lenguaje de programación Scheme con ejemplos de los requerimientos funcionales.

repositorio_21081166_LizamaGonzález.txt

Archivo de texto con el link del repositorio privado de GitHub con el proyecto.

Autoevaluación_21081166_LizamaGonzález.txt

Archivo de texto con la autoevaluación de cada requerimiento funcional y no funcional.

informe_21081166_LizamaGonzález.pdf

El presente informe.

Instrucciones de uso

Para crear los TDAs respectivos hay que respetar el dominio de las funciones de construcción presentes en el código respectivo de cada estructura:

TDA pixbit-d constructror:

(pixbit-d 0 0 1 5)

Creación de un pixbit-d con coordenadas x e y (0, 0), de color blanco y de profundidad 5

TDA pixrgb-d constructror:

(pixrgb-d 1 2 255 255 255 3)

Creación de un pixrgb-d con coordenadas x e y (1, 2), de color blanco y de profundidad 3

TDA pixhex-d constructror:

(pixhex-d 1 1 “#000000” 2)

Creación de un pixhex-d con coordenadas x e y (1, 1), de color negro y de profundidad 2

TDA image constructror:

(image 2 2 (pixbit-d 0 0 1 5) (pixbit-d 1 0 0 4) (pixbit-d 0 1 1 2) (pixbit-d 1 1 0 1))

Creación de una imagen 2x2 de tipo bitmap.

Resultados y autoevaluación

Se espera obtener un buen resultado sin fallos en el script de pruebas, sin tomar en cuenta los errores de formato presentados en el documento guía del proyecto del laboratorio, podría llegar a fallar alguna función si no se respetan los dominios y recorridos correspondientes.

Conclusiones del Trabajo

Utilizar el paradigma de programación funcional fue más complejo de lo que pensé por estar acostumbrado a la programación imperativa, pero después de programar bastante con este paradigma se me hizo más intuitivo el desarrollo del programa. Lo más difícil de esto fue la composición constante de funciones al no poder utilizar variables. A pesar de todo esto se logró realizar el desarrollo del software sin mayores complicaciones y se obtuvieron los resultados deseados.