Universidad de santiago de Chile

Labor Lætitia nostra

Informe para el ramo Paradigmas de programación

Laboratorio 1: Paradigma Funcional
Editor de imágenes usando Scheme, DrRacket

Sergio Osvaldo Andres Espinoza Gonzalez

Facultad de ingeniería

Agosto de 2022

Índice

1. Introducción

En este informe se abordará el laboratorio número 1 del ramo paradigmas de programación con el formato expuesto en el indice del mismo. A continuación se darán breves descripciones de los dos grandes pilares a ver para encontrar la posterior resolución.

1.1. Descripción del problema

Para empezar debemos conocer el problema a resolver, el cual es la creación de un software que permita la edición de imagenes de manera similar al software de código abierto GIMP o el de pago conocido como Photoshop.

1.2. Descripción del paradigma

Para la resolución del problema planteado anteriormente se hará uso de un lenguaje de programación llamado Racket el cual es un derivado de scheme que se encuentra enfocado en el paradigma de programación funcional, ya teniendo en cuenta esto podría surgirnos la pregunta "¿Que es el paradigma funcional?"

1.2.1. Paradigma funcional

El paradigma funcional es aquel que, como dice su nombre, hace uso de funciones y todo lo relacionado con las mismas, es decir, funciones de orden superior, evaluación de funciones en tiempo real, currificación y más.

2. Análisis del problema

Ya entendiendo por encima el problema y el paradigma a utilizar para solucionarlo profundizaremos en el mismo, ya que, conociendo bien el problema y sus requisitos especifícos llegaremos a una mejor solución.

1. TDAs: El primer desafio será diseñar los tipos de datos abstractos que nos permitan alcanzar la solución de la manera más eficiente posible, por lo que este primer requisito es más que nada de diseño de la solución.
2. Contructor de imágenes: Ya habiendo mencionado los TDAs el que principalmente se llevará nuestra atención será aquel que represente una imagen como tal, ya que, al implementarlo debemos tener diferentes consideraciones como:
   - Tipo de imagen: En un principio se deberían tener 3 tipos de imagen correspondientes a bitmap, pixmap y hexmax y que el programa pueda reconocer de que tipo se trata; la implementación de lo antes mencionado se verá en la siguiente sección.
   - Debe ser posible también poder cambiar entre imágenes tipo pixmap y hexmap (RGB y hexadecimal).
   - ¿Está comprimida?: Es necesario si una imagen está o no comprimida
   - Comprimir y descomprimir: Como ya se dijo que una imagen puede estar o no comprimida entonces deben haber funciones que compriman y descompriman la imagen como tal, dependiendo de la implementación podría verse también como requisito una función que nos diga cuál es el color más usado en la imagen (Histograma).
3. Cambiar dirección o rotar: Otro requisito especifífico a abordar es dar la posibilidad de invetir la imagen ya sea en el eje X o Y, además de poder rotar esta en 90° hacia la derecha o izquierda.
4. Recortar: También se debe dar la posibilidad de recortar la imagen para obtener una nueva imagen de un subsector de la original.
5. Invertir colores: Como el nombre lo dice, se debe ser capaz de inverit los colores independiente del tipo de imagen.

6. Ajustar canal: Debe también darse la posibilidad de ajustar el canal de una imagen con pixeles RGB-D (con profundidad).
7. Imagen a string: Debe también cubrirse la posibilidad de que la imagen pueda ser representada como un conjunto de caracteres.
8. Separar en capas: La imagen debe tener profundidad, y para esto, se deben tener varias capas entre las que le es posible separarse.

3. Diseño de la solución

Comenzando ahora con el diseño de la solución para la problemática de este semestre deberiamos partir definiendo que tipos de dato abstracto o TDAs, los cuales serán los que nos permitirán una mejor representación para este editor de imagenes.

3.1. TDAs e implementación

El objeto central con el que trabajaremos en un editor de imagenes es bastante predecible, las imagenes, pero a su vez bien es sabido que las imagenes están subdivididas en pixeles con los que trabajar si se quieren realizar cambios en esta; así que, para este trabajo definiremos 3 tipos de pixeles y su posterior posible implementación para que luego podamos trabajar correctamente las imagenes en un archivo principal.

Los pixeles en sí estarán un poco más adelante, pero algo que los 3 tipos de estos tendrán en común, es que serán implementados con listas donde el segundo y tercer elemento corresponden a la posición (x,y) del pixel y el último elemento de la lista corresponderá a la profundidad del pixel, donde, estos 3 valores mencionados son enteros positivos.

La gran diferencia se encontrará en los datos que hay entre medio de los valores (x,y) y depth (profundidad), además del primer elemento el cual será un identificador de tipo de pixel (1,2 o 3); los pixeles quedarán de la siguiente forma:

- Pixbit-d (1): Este tipo de pixel tendrá un bit entre medio de tipo entero, el cual solo podrá tomar los valores 0 y 1. De esta forma un pixbit-d será una lista de la forma:

$$'(int, int, int, int, int) = (1, x \geq 0, y \geq 0, bit(0|1), depth \geq 0)$$

- Pixrgb-d (2): Este tipo de pixel tendrá además tendrá 3 enteros entre 0 y 255 que representarán la cantidad de cada color que tiene el pixel siendo r = rojo (red), g = verde (green) y b = azul (green) quedandonos una lista de la forma:

$$'(int, int, int, int, int, int, int) = (2, x \geq 0, y \geq 0, 0 \leq r \leq 255, 0 \leq g \leq 255, 0 \leq b \leq 255, depth \geq 0)$$

- Pixhex-d (3): Este tipo de pixel tendrá un string con formato #XXXXXX donde las X pueden tomar valores de un dígito hexadecimal (entre 0 y F(15)) donde las 2 primeras X es la cantidad de rojo, las 2 de al medio son de verde y las 2 últimas son de azul; la representación quedaría de la forma:

$$'(int, int, int, string, int) = (3, x \geq 0, y \geq 0, hex, depth \geq 0)$$

presentar su enfoque de solución, describir, diagramar, descomposición de problemas, algoritmos o técnicas empleados para problemas particulares, recursos empleados

4. Consideraciones de implementación

El código del proyecto estará en la carpeta Código, donde se encontrará el main con las funciones principales del enunciado; en esta carpeta se encontrará también otra carpeta que contiene los TDAs mencionados en la sección anterior "TDAs". No se usará ninguna biblioteca externa al proyecto y el compilador usado será el incluido en DrRacket.

5. Instrucciones de uso

(ejemplos, resultados esperados, posibles errores) (max 1 página. Complementar con ejemplos detallados en anexos que están fuera del límite de 10 páginas) (15 %)

6. Resultados obtenidos

Esto puede abordarse listando todos los requerimientos del proyecto en una tabla indicando el grado de alcance. Indicar que tipos de pruebas se hicieron. Cuantas de las pruebas fueron exitosas, cuantas fracasaron, razones de fallos. Especificar que funciones no se completaron y el porque no se completaron. (máximo 1 página) (10 %)

7. Evaluación completa

(Einstein, 1905)

8. Conclusión

(respecto de los alcances, limitaciones, dificultades de usar el paradigma para abordar el problema. Para los informes 2 en adelante, contrastar resultados alcanzados con el paradigma de turno con los anteriores) (máximo 1 página) (10 %)

Referencias

Einstein, A. (1905). Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]. *Annalen der Physik*, *322*(10), 891–921. doi: http://dx.doi.org/10.1002/andp.19053221004

Anexos

a