UNIVERSIDAD
DE SANTIAGO
DE CHILE

INFORME DE LABORATORIO 1: SIMULACIÓN DE EDITOR DE IMÁGENES EN SCHEME

Nombre: Diego Antonio Armijo Palominos

Profesor: Victor Flores

Fecha: 26/09/22

INTRODUCCIÓN

La programación es solo el proceso de crear una serie de acciones cronológicas para cumplir un objetivo deseado. Estas instrucciones le dicen a una computadora como realizar algún tipo de tarea. Un lenguaje de programación es la herramienta para automatizar informaciones y acciones a través de una computadora. Los lenguajes de programación más conocidos son: Basic (1964), C++ (1983), Python (1991), Java (1995), C# (2000), entre otros. Cada uno de estos lenguajes permite afrontar un problema desde una perspectiva diferente. Para abordar este laboratorio se utilizará el paradigma funcional, el cual utiliza el lenguaje de programación Scheme a través del compilador Dr. Racket. Este paradigma se basa en el uso de funciones para llegar a una solución, lo que quiere decir que no existen variables ni ciclos.

Este informe comienza describiendo el problema, luego una descripción del paradigma, los objetivos que se esperan lograr, el análisis del problema, diseño de la solución, los aspectos de implementación, instrucciones y ejemplos de uso, resultados, autoevaluación y una conclusión al proyecto.

DESCRIPCIÓN DEL PROBLEMA

Se pide desarrollar un programa tipo “Photoshop” que permita crear una imagen y hacerle modificaciones. Esta imagen se rige por ciertas propiedades que debemos tener en cuenta:

Image: En base a ciertos parámetros, se crea una imagen que contiene el ancho, alto y los pixeles de la imagen.

Pixbit-d: Tipo de píxel que acepta la imagen, este es de tipo bit.

Pixrgb-d: Tipo de píxel que acepta la imagen, es de tipo rgb.

Pixhex-d: Tipo de píxel que acepta la imagen, es de tipo hexadecimal**.**

DESCRIPCIÓN DEL PARADIGMA

La programación funcional es una forma en la cual podemos resolver diferentes problemáticas. Este paradigma trabaja principalmente con funciones, donde se evitan los datos mutables. Dentro de este paradigma se destaca:

Funciones de orden superior: Funciones que pueden recibir como entrada una función o bien dar resultado otra función.

Funciones anónimas: Provienen del cálculo lambda y se pueden expresar sin definirle un nombre. Se suelen utilizar en funciones propias del lenguaje como filter o map.

Composición de funciones: Operación donde dos funciones generan una tercera.

Recursividad: Proceso mediante el que una función se llama a sí misma de forma repetida, hasta que se satisface alguna determinada condición.

Currificación: Consiste en transformar una función que utiliza múltiples argumentos en una secuencia de funciones que utilizan un único argumento.

OBJETIVOS

El objetivo principal del proyecto es aprender sobre el paradigma y la programación funcional. Esta te enseña ver un problema desde un enfoque distinto al que se está acostumbrado, con diferentes herramientas para afrontarlo.

DESARROLLO

ANALISIS DE PROBLEMA

Antes de generar una imagen tendremos que crear los pixeles de esta. Cada píxel tiene sus reglas y condiciones, es porque eso hay que verificar que los parámetros que ingresa el usuario sean valores que estén dentro del rango aceptado por las condiciones del píxel.

Para poder generar una imagen es necesario saber las dimensiones de esta, ya que en base a estas tendremos que verificar que las coordenadas de cada píxel que ingrese el usuario sean correctas (estén en su posición) y que las dimensiones sean acordes al tamaño de la lista de pixeles. Ya verificado todo podremos generar la imagen y manipularla según se nos pida.

Pixbit-d: Corresponde a un conjunto de asociaciones, donde un usuario genera un píxel de tipo bit. En Scheme se puede representar como una lista que contiene las coordenadas del píxel, el bit y la profundidad (posX(int) x posY(int) x bit [0-1] x Depth(int)).

Pixrgb-d: Corresponde a un conjunto de asociaciones, donde un usuario genera un píxel de tipo rgb. En Scheme se puede representar como una lista que contiene las coordenadas del píxel, su color (R G B) y la profundidad (posX(int) x posY(int) x R(C) x G(C) x B(C) x D(int)).

Pixhex-d: Corresponde a un conjunto de asociaciones, donde un usuario genera un píxel de tipo hexadecimal. En Scheme se puede representar como una lista que contiene las coordenadas del píxel, su color representado de forma hexadecimal y la profundidad (posX(int) x posY(int) x Hex(string) x D(int)).

Image: Corresponde a un conjunto de asociaciones, donde un usuario genera una imagen. En Scheme se puede representar como una lista que contiene las dimensiones de la imagen (Width – Height) y una lista con los pixeles (Width(int) x Height(int) x [pixbit-d* | pixrgb-d* | pixhex-d*]). Cabe destacar que image es una función de tipo variadic que puede recibir n-entradas como argumento, además, es necesario tener los creadores de pixbit-d | pixrgb-d | pixhex-d para que la imagen pueda ser creada.

Cada TDA se separa en un archivo único. Estos están conectados al TDA principal (image). La forma de exportar-importar funciones se hace mediante las funciones require y provide.

DISENO DE SOLUCION

Para el diseño de solución utilizamos los creadores de tipos de datos específicos (TDA), donde primero ira la representación, constructores, funciones de pertenencia, selectores, modificadores y por últimos otras funciones. Para la solución implementamos cuatro TDAs:

TDApixbit-d:

Representación: Cuando el usuario ingrese los parámetros y la función respectivamente, este retornara en una lista con los elementos del píxel.

Constructor: Dado las coordenadas, el bit y la profundidad, este genera un píxel bit valido, el cual es una lista con los elementos de entrada y algunos adicionales.

Función de Pertenencia, Selectores, Modificadores y Otras Funciones (ver ...).

TDApixrgb-d

Representación: Cuando el usuario ingrese los parámetros y la función respectivamente, este retornara en una lista con los elementos del píxel.

Constructor: Dado las coordenadas, el color rgb y la profundidad, este genera un píxel rgb valido, el cual es una lista con los elementos de entrada y algunos adicionales.

Función de Pertenencia, Selectores, Modificadores y Otras Funciones (ver ...).

TDApixhex-d

Representación: Cuando el usuario ingrese los parámetros y la función respectivamente, este retornara en una lista con los elementos del píxel.

Constructor: Dado las coordenadas, el color hexadecimal y la profundidad, este genera un píxel hex valido, el cual es una lista con los elementos de entrada y algunos adicionales.

Función de Pertenencia, Selectores, Modificadores y Otras Funciones (ver ...).

TDAimage (principal)

Representación: Cuando el usuario ingrese los parámetros y la función respectivamente, este retornara en una lista con los elementos de una imagen valida.

Constructor: Dada la altura, ancho y los pixeles, este generará una imagen valida, la cual contendrá los elementos de entrada y algunos adicionales.

Función de Pertenencia, Selectores, Modificadores y Otras Funciones (ver ...).

Las operaciones obligatorias se muestran en ….

ASPECTOS DE IMPLEMENTACION

COMPILADOR

Para el desarrollo de este proyecto es necesario tener instalado el compilador Dr. Racket (6.11). Se pueden utilizar todo tipo de funciones de Scheme y Racket, menos las que simulan el uso de variable (ejemplo: let!).

La implementación está orientada en las listas, la cual tiene funciones como CAR, CDR, entre otras. Se espera no trabajar directamente con el nombre de estas (encapsularlas).

ESTRUCTURA DEL CODIGO

Todos los TDAs siguen la misma estructura. El Código comienza con el constructor, las funciones de pertenencia, los selectores, los modificadores y las otras funciones.

INSTRUCCIONES DE USO

Para el correcto funcionamiento del programa, es necesario tener todos los archivos dentro de una carpeta, ya que, de no tenerlos, el archivo “TDAimage” no se podrá ejecutar.

En Dr. Racket se compila el programa mediante el botón “run”. Cada TDA tiene ejemplos de uso de cada función. También se agregó un script de pruebas para probar todas las funciones.

RESULTADOS ESPERADOS

Se espera poder crear una imagen valida, a la cual se le puedan aplicar funciones (modificadores) para simular el uso de un editor de imágenes. Todo esto mediante el paradigma funcional y los recursos que nos aporta (trabajar con listas, recursión, composición de funciones, currificación, etc.)

POSIBLES ERRORES

Durante el desarrollo del proyecto, todas las funciones arrojaron el resultado esperado, de igual manera todas las funciones están documentadas. Cabe destacar que se lograron – de 20 funciones en total.

RESULTADOS Y AUTOEVALUACIÓN

RESULTADOS OBTENIDOS

Todos los resultados fueron los esperados, ya que se logró crear imágenes de distinto tamaño y modificarlas de manera correcta. Se realizaron múltiples pruebas que pueden ver en el script de prueba.

AUTOEVALUACION

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

CONCLUSIÓN

Se puede concluir que se cumplió con los requerimientos básicos requeridos (funciones) en el proyecto. También se aprendió a utilizar de forma correcta el paradigma funcional ayudándonos a ver los problemas desde otra perspectiva y con otras herramientas para afrontar la solución. Si bien no se lograron completar los requerimientos en su totalidad, se completó la gran mayoría, y las funciones que se completaron arrojan los resultados correctos

ANEXOS

Tabla N ° 1: Autoevaluación de los requerimientos funcionales.

Requerimientos Funcionales	Evaluación
TDAs	0.75
TDA image - constructor	1
TDA image - bitmap?	1
TDA image - pixmap?	1
TDA image - hexmap?	1
TDA image - compressed?	1
TDA image - flipH	1
TDA image - flipV	1
TDA image - crop	1
TDA image - imgRGB->imgHex	1
TDA image - histogram	0.25
TDA image - rotate90	1
TDA image - compress	0
TDA image - edit	0
TDA image - invertColorBit	1
TDA image - invertColorRGB	0
TDA image - adjustChannel	0
TDA image - image->string	0
TDA image - depthLayers	0
TDA image - decompress	0

Tabla N ° 2: TDApixbit-d.

Tipo de función	Nombre	Descripción
Funcion de pertenencia	pixbit-d?	Verifica si el pixbit-d es un pixbit-d valido.
Constructor	pixbit-d	Construye un pixbit-d
Selector	bit->getCoord	Obtiene una lista con las coordenadas de un bit.
Selector	bit->getBit	Obtiene el valor de un bit (0|1).
Selector	bit->getDepth	Obtiene la profundidad de un bit.
Selector	bit->getType	Obtiene el tipo de un bit.
Selector	bit->getOpuesto	Obtiene el bit opuesto de un bit.

Tabla N ° 3: TDApixrgb-d.

Tipo de función	Nombre	Descripción
Función de pertenencia	pixrgb-d?	Verifíca si el pixbit-d es un pixrgb-d valido.
Constructor	pixrgb-d	Construye un pixrgb-d
Selector	rgb->getRGB	Obtiene el color rgb de un pixrgb
Selector	rgb->getR	Obtiene el color r de un pixrgb.
Selector	rgb->getG	Obtiene el color g de un pixrgb.
Selector	rgb->getB	Obtiene el color b de un pixrgb.
Selector	rgb->getHex	Obtiene el color hexadecimal de un pixrgb.
Selector	rgb->getCoord	Obtiene las coord de un pixrgb
Selector	rgb->getDepth	Obtiene el d de un pixrgb
Otras funciones	transform1	Trasformar un numero rgb a hex (parte 1).
Otras funciones	Transform2	Trasformar un numero rgb a hex (parte 2).
Otras funciones	append-transform num	Función que une la parte 1 y 2 de transformar un numero rgb a hex
Otras funciones	append-rgb->hex rgb	Función que transforma un rgb a hex

Significados. (2016, December). *Significado de Programación*. Significados; Significados. https://www.significados.com/programacion/

Recursividad. (2022). Ccia.ugr.es. https://ccia.ugr.es/~jfv/ed1/c/cdrom/cap6/cap66.htm#:~:text=Definici%C3%B3n%3A%20Se%20llama%20recursividad%20a,determina%20mediante%20un%20resultado%20anterior.