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

INFORME DE LABORATORIO 2:

GIMP MEDIANTE PROLOG

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

- **Fecha:** 02 de noviembre 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 de programación Prolog, basándose en la particularidad de este, el paradigma lógico, de esta manera se mostrara cómo se abordó el tema, el enfoque que se le dio y como se llegó a una solución que permitió realizar un programa con todos los requisitos básicos completados en su totalidad, además se dará una 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 lógica, como su nombre lo dice, trabaja en términos de la lógica matemática de primer orden, lo que implica que se trabaje sobre valores Booleanos, es decir, sobre verdadero y falso.

Este paradigma trabaja con una base de datos, con hechos y reglas que se aplican bajo la lógica, luego se puede hacer consultas sobre esta base de conocimientos.

Para este trabajo se considera el lenguaje SWI-Prolog, el cual es un lenguaje de programación interpretado y netamente del paradigma lógico, lo que lo hace ser una buena herramienta en el desarrollo de este software.

La programación lógica tiene tres mecanismos básicos que explicaremos a continuación:

- **La unificación:** Es el mecanismo utilizado por Prolog para resolver una meta, la cual reemplaza términos para encontrar soluciones que sean equivalentes.
- **Backtracking automático:** Si no se logra la unificación, este algoritmo hace una vuelta hacia atrás, probando de otra forma lograr esta unificación.
- **Estructuras de datos basadas en árboles:** Al hacer Backtracking, se genera una búsqueda en espacio de estados, lo que provoca que estas soluciones se vayan buscando en una estructura en forma de árbol.

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 lógico se enfoca en el resultado y no en el procedimiento, lo que implica un mayor nivel de abstracción.

OBJETIVOS

El objetivo del proyecto se basa en aprender y utilizar los conceptos del paradigma lógico, como también el uso del lenguaje de programación Prolog. 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 similar a GIMP o Adobe Photoshop, en el lenguaje Prolog, que tiene como propósito llevar a cabo los conocimientos vistos en clases y la correcta utilización del material, 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 los predicados de pertenencia y modificación para una imagen, pueden variar dependiendo del tipo de pixel, por lo que un solo predicado o regla principal, tendría que ser capaz de validar una acción por igual para los 3 tipos de imágenes, por ende, para cada regla 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 predicados 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, 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 Prolog y se decide usar el interpretador SWI-Prolog, ya que es una herramienta bastante útil al momento de hacer el bracktracking y con una interfaz bastante simple a la hora de hacer consultas. 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 usa el comando use_module("nombreArchivo") para importar en prolog de esta manera obtenemos todos los predicados, hechos, reglas, de cada archivo importado. Para este programa, se estructuran 7 archivos: "main.pl", "FuncionesBase.pl", "TDA_image.pl", "TDA_pixbit.pl", " TDA_pixhex.pl", " TDA_pixrgb.pl" y el "prueba.pl" (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 selectores y 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 los predicados en el archivo "main.pl". Luego de 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.

"((pixbit(0, 0, 1, 10, PA), pixbit(0, 1, 0, 20, PB), pixbit(1, 0, 0, 30, PC), pixbit(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I))."

En caso de querer aplicar directamente las otras funcionalidades, puede ingresar el mismo código anterior y solo agrega el predicado al final de la consulta; como por ejemplo:

"(pixbit(0, 0, 1, 10, PA), pixbit(0, 1, 0, 20, PB), pixbit(1, 0, 0, 30, PC), pixbit(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), imageFlipH(I , I2), imageFlipH(I2 , I3))."

Y asi sucesivamente con todos los predicados:

"imagenIsBimap(I)."
"imagenIsHexmap (I)."
…

RESULTADO ESPERADO

Se espera que el programa sea 100% lógico para cumplir con el paradigma estudiado y 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 predicados, hechos, reglas, etc.

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	0.75
MAIN	0.5

En el archivo main todos los predicados que existentes funcionan al 100% sin embargo no esta el total de los predicados solicitadas en el proyecto es por esta razón que la asignación es de 0.5 y 0.75 en el Main y el TDA Image, 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 segundo laboratorio fue el esperado ya que se pudo realizar todas los predicados básicos para cada TDA, como los son los constructores, selectores, modificadores y pertenencia. Sin embargo, se requiere reforzar más en alcanzar a hacer los

predicados 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 predicado, cabe destacar que hubo muchos errores, pero ninguno que haya tomado un tiempo significativo a la hora de solucionarlo.

Finalmente, en comparación al laboratorio 1, en este se obtuvo un mayor aprendizaje y compresión del paradigma, por lo mismo se logró abarcar mayores cantidades de requisitos solicitados, por lo que si hubo una mejora significativa en comparación al anterior.

4. REFERENCIAS

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