Pontificia Universidad Javeriana

Entrega #1 Proyecto

Angel Eduardo Morales Abril
Tomas Alejandro Silva Correal
Juan Guillermo Pabón Vargas
Gabriel Jaramillo Cuberos
Salomon Avila

Estructuras de datos

Jhon Franco Corredor

05 marzo 2025

Indice

Introducción

Para la entrega número uno de este proyecto, se solicitó la planificación, diseño, desarrollo e implementación de la función para proyección 2D del proyecto. El objetivo de este método es el de convertir un volumen de imágenes en formato pgm a una sola imagen que proyecta los valores que el usuario solicita. Se inicia el proceso de diseño creando los TAD y el diagrama de relación de las clases que se utilizan para el método, con tal de asegurar que se cuente con los recursos necesarios para su posterior implementación. Tras diseñar los TAD y el diagrama se crearon una serie de pasos de forma simple para facilitar la programación del método, con lo que se generó la implementación y un plan de pruebas correspondiente al funcionamiento del mismo, todos estos elementos se mostrarán a lo largo de este documento.

TAD

Clases a utilizar: Imagen.cpp y VolumenImagenes.cpp

TAD Imagen

Datos minimos:

- vecImagen: vector de vectores de enteros, representa cada píxel de la imagen.
- maxClaro: entero, representa el máximo de la intensidad de un pixel.
- dimensionX: entero, define el ancho de la imagen.
- dimensionY: entero, define la altura de la imagen.
- formato: cadena de caracteres, dice el tipo de formato de la imagen.
- nombre: cadena de caracteres, dice el nombre de la imagen.

Operaciones:

- getVecImagen(): retorna el vector de vectores de vecImagen.
- setVecImagen(vecImg): asigna a vecImagen un vector de vectores de enteros.

- getMaxClaro(): retorna el valor máximo de claridad (brillo) permitido en la imagen.
- setMaxClaro(maximo): asigna al valor de “maxClaro” un valor entero que entra de parámetro.
- getDimensionX(): retorna el entero de dimensionX.
- setDimensionX(dimension): asigna al valor de “dimensionX” un valor entero que entra de parámetro.
- getDimensionY(): retorna el entero de DimensionY.
- setDimensionY(dimension): asigna al valor de “dimensionY” un valor entero que entra de parámetro.
- getFormato(): retorna el string de Formato.
- setFormato(form): asigna al valor de “Formato” un string que entra de parámetro.
- getNombre(): retorna el string de Nombre.
- setNombre(nombre2): asigna al valor de “Nombre” un string que entra de parámetro.
- cargarDesdePGM(ruta): carga imagen .pmg usando una ruta de archivo.
- guardarComoPGM(ruta): guarda imagen como .pmg a una ruta específica.
- mostrarInfo(): muestra la información relacionada a la Imagen.

TAD VolumenImagenes

Datos mínimos:

- cantidadImagenes: entero, define la candida de imágenes que se van a cargar
- imagenes: vector de imágenes, contiene las imágenes ordenadas según el orden de la imagen en el archivo
- nombreBase: cadena de caracteres, nombre base de la colección de imágenes

Operaciones:

- getCantidadImagenes(): retorna el número de cantidad de imágenes
- getImagenes(): retorna el vector de imágenes
- setCantidadImagenes(cantidad): asigna al valor de “cantidadImagenes” un valor entero que entra de parámetro.
- setImagenes(imagen): asigna al vector de vectores “Imagenes” un vector de mismo tipo que entra como parámetro de entrada.
- setNombre(nom_base): asigna un nombre a la colección de imágenes.
- getNombre(): retorna el nombre de la colección de imágenes.
- cargarDesdeArchivos(base, numImagenes): Carga las imágenes desde la ruta e indica la cantidad de imágenes que se cargaron.
- mostrarInfo(): Muestra la información relacionada a la base y el número de imágenes cargadas.
- GenerarProyeccion2D(dirección, criterio, rutaFinal): Se genera una proyección 2D del conjunto de imágenes dado la dirección de vista y el criterio especificado.

Diagrama de relación:

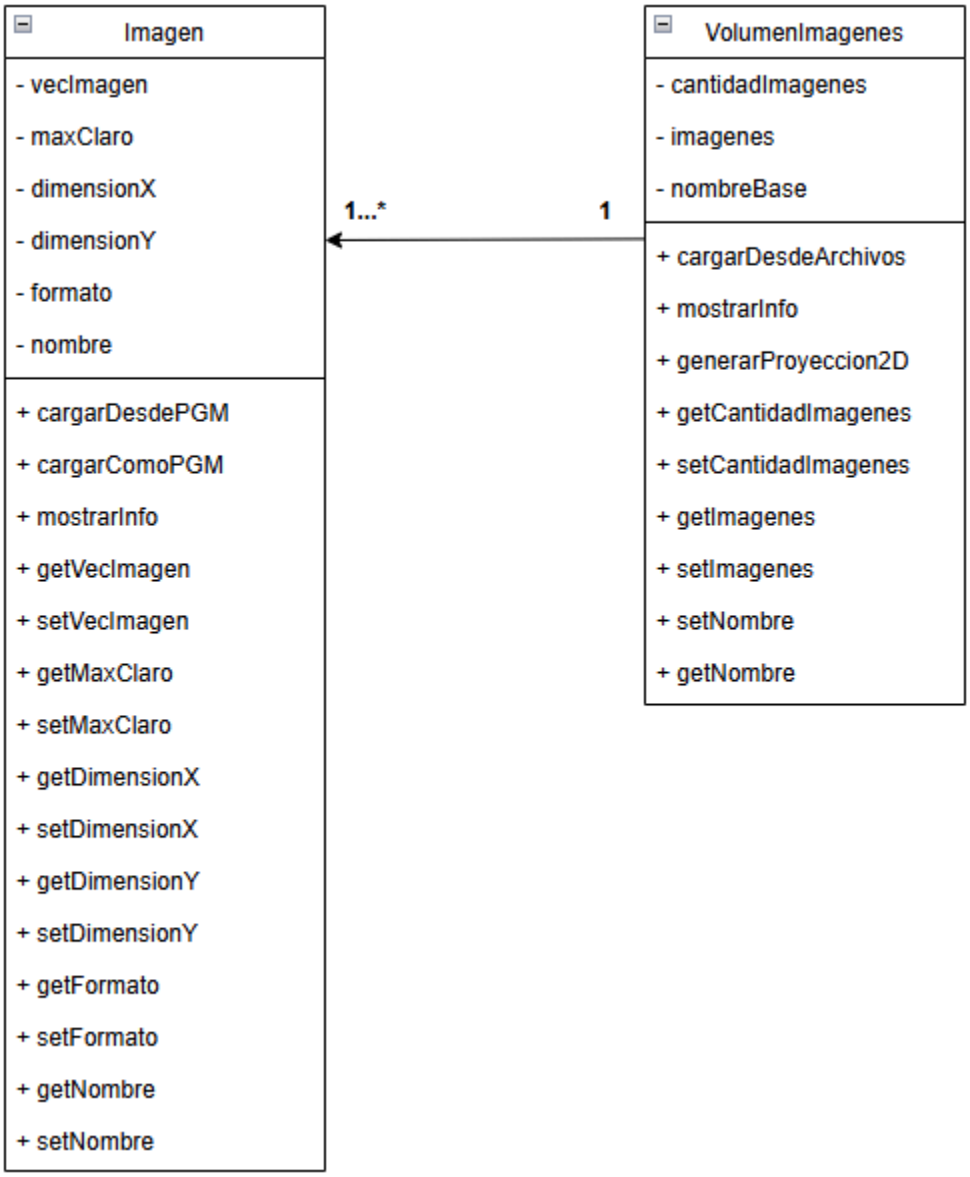

Imagen 1, Diagrama de relación entre Imagen y VolumenImagenes

Pasos a seguir: generarProyeccion2D

1. Tomar la serie ordenada de imágenes de la memoria, junto a la dirección (x,y o z) y el criterio (mínimo, máximo, promedio y mediana)
2. Cargar el vector de vectores de cada imagen tras cargar el archivo pgm dentro del vector cargado en memoria.

3. Dada la dirección y el criterio, calcular el valor del criterio (según los valores de la imagen y el criterio elegido), viajando en torno a la dirección para el cálculo correspondiente (recorriendo por los valores de vector e imágenes necesarias).
4. Agregar los valores que cumplan con el criterio (nivel de intensidad) al vector de vectores que será la imagen resultado.
5. Crear una nueva imagen y asignarle el vector de imágenes, junto con las dimensiones y valores correspondientes (el formato siendo el mismo que el de las imágenes cargadas).
6. Guardar la imagen como archivo .pgm

Plan de pruebas:

Esta entrega del proyecto propone una solución al problema de la proyección de imágenes 3D en planos 2D, enfocándose en la simulación del proceso de captura de radiografías.

Para ello, se utilizará el sistema desarrollado en la primera entrega del proyecto, ampliándolo con la definición de funcionalidades, tales como:

- Carga de imágenes desde archivos **PMG**.
- Carga de volúmenes de imágenes desde **PMG**.
- Almacenamiento y visualización de información de imágenes.
- Almacenamiento y visualización de información de volúmenes de imágenes.
- Proyección de imágenes 3D en un plano 2D.

Con el fin de garantizar el correcto funcionamiento de estas funcionalidades, se diseñará un plan de pruebas específico para cada una de ellas:

- Carga de imágenes desde archivos **PMG**.

Prueba	Resultado esperado	Resultado real
cargar_imagen nombre_imagen.pgm	La imagen imagen.pgm ha sido cargada.	
cargar_imagen nombre_imagen.png	La imagen imagen.pgm no ha podido ser cargada.	
cargar_imagen	Falta de argumentos.	
cargar_imagen imagen_inexistente.pgm	La imagen imagen.pgm no ha podido ser cargada.	
cargar_imagen nombre_imagen.pgm (después de haber cargado otra imagen)	La imagen imagen.pgm ha sido cargada.	

	(Sobreescribe la anterior)	

- Carga de volúmenes de imágenes desde **PMG**.

Prueba	Resultado esperado	Resultado real
cargar_volumen volumen 10	El volumen “volumen” ha sido cargado.	
cargar_volumen volumen -5	La cantidad de imágenes no es válida.	
cargar_volumen volumen	Falta de argumentos.	
cargar_volumen volumen 200	La cantidad de imágenes supera el límite permitido.	
cargar_volumen volumen 10 (con una imagen faltante)	El volumen “volumen” no ha podido ser cargado.	
cargar_volumen volumen 10 (después de haber cargado otro volumen)	El volumen “volumen” ha sido cargado. (Sobreescribe el anterior)	

- Almacenamiento y visualización de información de imágenes.

Prueba	Resultado esperado	Resultado real
Info_imagen (con imagen cargada)	Imagen cargada en memoria: imagen.pgm, ancho: W, alto: H.	
Info_imagen (sin imagen cargada)	No hay una imagen cargada en la memoria.	

- Almacenamiento y visualización de información de volúmenes de imágenes.

Prueba	Resultado esperado	Resultado real
Info_volumen (con volumen cargado)	Volumen cargado en memoria: volumen, tamaño: n_im, ancho: W, alto: H.	

Info_volumen (sin volumen cargado)	No hay un volumen cargado en memoria.	

- Proyección de imágenes 3D en un plano 2D.

Prueba	Resultado esperado	Resultado real
proyeccion2D x maximo salida.pgm (con volumen cargado)	La proyección 2D del volumen en memoria ha sido generada y almacenada en el archivo salida.pgm.	
proyeccion2D a maximo salida.pgm	El volumen aún no ha sido cargado en memoria.	
proyeccion2D z promedio salida.pgm (sin volumen cargado)	Dirección no válida. Las opciones son: x, y, z.	
proyeccion2D x suma salida.pgm	Criterio no válido. Las opciones son: minimo, maximo, promedio, mediana.	
proyeccion2D x maximo	Falta de argumentos.	

Conclusiones:

- Integración de conocimientos teóricos y prácticos: el proyecto hizo posible la aplicación de los conceptos de estructuras de datos y algoritmos aprendidos, mostrando cómo la teoría de las estructuras y los tipos abstractos de datos se convierten en soluciones prácticas para el procesamiento de imágenes.

- Modularidad y escalabilidad: la división del sistema en componentes muestra la importancia del diseño de soluciones haciendo uso de módulos, facilitando el mantenimiento y la ampliación del sistema, además de la creación de librerías para el acceso a métodos y funciones de forma eficiente.

- Eficiencia en el manejo de datos: la implementación del sistema para el procesamiento de imágenes necesitó utilizar estructuras de datos, tanto para almacenar grandes volúmenes de información como para realizar operaciones complejas.

- Cambios respecto a la entrega anterior: pese a ideas originales, y por ámbitos de complejidad y tiempo, se optó por eliminar la estructura o modelo MVC (modelo vista-controlador), con tal de permitir correcciones más sencillas y modificaciones ágiles para el desarrollo del programa.

- Importancia de la interacción entre el usuario y el sistema: el desarrollo de una interfaz de línea de comandos muestra cómo una interacción clara y amigable puede mejorar la experiencia del usuario al permitirle ejecutar y comprender los diferentes procesos de transformación y análisis de imágenes.