Tecnológico de Monterrey

INSTITUTO TECNOLÓGICO Y

DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS TOLUCA

Análisis y diseño de algoritmos avanzados (Gpo 601)

Documentación de Proyecto Delaunay

Profesor: Dr. José María Aguilera Méndez

Equipo 6

Ángel Rogelio Cruz Ibarra - A01771434

Ana Karen Toscano Díaz - A01369687

Liliana Camacho Suárez - A01369125

Fecha de entrega: 25 de noviembre de 2024

Link de repositorio de GitHub: https://github.com/toscanodiaz/Delaunay.git

DESCRIPCIÓN GENERAL

Este proyecto tiene como objetivo generar una triangulación de Delaunay, una técnica geométrica utilizada ampliamente en computación gráfica, lo que ayuda en la modelación de terrenos, simulaciones físicas e incluso llegando al terreno del reconocimiento facial.
En este caso nosotros trabajamos sobre una superficie costera delimitada, el objetivo era implementar un programa que genere una triangulación a partir de un conjunto de puntos proporcionados en un archivo **.poly** y que genere archivos de salida **.node** y **.ele** con los resultados.
Hicimos uso del algoritmo de Bowyer-Watson, lo que crea una malla triangulada que respeta la forma del perímetro costero, lo que ajusta la densidad de los nodos según la proximidad a la costa o a los bordes del mapa, utilizamos el método "Divide y Vencerás", lo que favorece el conjunto de puntos, maximizando el ángulo mínimo de todos los triángulos.

Entrada de datos

- Imagen en formato .jpg de cualquier superficie costera.
- Número de puntos: Cantidad total de puntos definidos.
- Coordenadas de los puntos: Cada punto tiene valores **X** y **Y**
- Puede incluir identificadores adicionales como etiquetas o restricciones para los puntos.

Características del programa

- Dependencias de librerías utilizadas:
    - <studio.h>: Para entrada/salida de datos.
    - <stdlib.h>: Para manejo de memoria dinámica.
    - <math.h>: Para cálculos matemáticos avanzados.
- Generar vértices (nodos) a lo largo de la costa dentro del perímetro delimitado:
    - Mayor densidad de puntos en la costa.
    - Menor densidad de puntos en los bordes del mapa (donde no hay tierra).
- Realizar triangulaciones de Delaunay basadas en el algoritmo de Bowyer-Watson.
- Crear nuevos puntos al intersectar las líneas de los triángulos.

BASE TEÓRICA

La triangulación de Delaunay es una técnica geométrica que subdivide un conjunto de puntos en triángulos de manera que se cumple la condición de Delaunay: el círculo circunscrito de cualquier triángulo de la triangulación no contiene otros puntos del conjunto en su interior. Esto hace que los triángulos resultantes sean lo más "equiláteros" posible, maximizando los ángulos mínimos de la malla.

Algoritmo de Bowyer-Watson

El algoritmo de Bowyer-Watson es un método incremental para generar una triangulación de Delaunay, funciona de la siguiente manera:

- *Triángulo Superpoblador*: Se inicia con un triángulo suficientemente grande que cubre todos los puntos del conjunto.
- *Inserción de Puntos*: Cada punto del conjunto se añade uno por uno.
- *Círculo Circunscrito*: Se verifica si el punto rompe la condición de Delaunay.
- *Reconfiguración*: Si el punto está dentro del círculo circunscrito de algún triángulo, se eliminan estos triángulos y se reconfiguran con nuevas aristas que incluyen el nuevo punto.
- *Terminación*: Se eliminan los triángulos que tocan los vértices del triángulo superpoblado inicial, dejando únicamente la triangulación deseada.

Puntos de Steiner
Los puntos de Steiner son puntos adicionales que se insertan estratégicamente en una triangulación para mejorar su calidad. En el contexto de la triangulación de Delaunay, estos puntos son utilizados principalmente para mejorar la calidad de los triángulos, eliminar ángulos muy agudos y optimizar la forma de los elementos en la malla. Se colocan típicamente en el circuncentro de triángulos de mala calidad.

Densidad de Nodos
Para representar fielmente una superficie costera, la densidad de nodos debe ajustarse a las características geométricas del área:
- *Costa*: Alta densidad para capturar los detalles.
- *Bordes del mapa*: Baja densidad para mantener simplicidad donde no hay detalles relevantes.

FUNCIONALIDAD

1. *Carga de la imagen*

El usuario proporciona una imagen en formato .jpg que representa la región costera.

2. *Delimitación del perímetro*

El programa lee la imagen .jpg y transforma las coordenadas de píxeles en coordenadas geográficas ajustadas al área delimitada.

3. *Generación de nodos*

El usuario, manualmente, posiciona los nodos dentro del área delimitada.

4. *Triangulación*

El programa implementa el algoritmo de Bowyer-Watson para construir la triangulación.
Inicio: El programa comienza con un triángulo que cubre todos los nodos.
Iteración: Cada nodo nuevo se inserta y se verifica si rompe la condición de Delaunay (el círculo circunscrito de un triángulo no debe contener ningún nodo en su interior).
Actualización: Se ajustan los triángulos y se agregan nuevos nodos en las intersecciones de las líneas

5. *Salida de archivos*

.poly: Lista inicial que contiene información sobre nodos, coordenadas, segmentos y agujeros.
.ele: Información de los triángulos, incluyendo índices de los nodos que los forman.
.node: Nodos generados tras la triangulación.

.exe: Ejecutable que reproduce el programa.

6. *Visualización Gráfica*

MATLAB

Lo utilizamos para representar los puntos y triángulos de la triangulación, lo que permite analizar visualmente los resultados generados en el programa, y con el postprocesamiento, se analizan las propiedades como los ángulos, longitudes de arista o calidad de los triángulos.

SWAN

Este fue utilizado para la simulación numérica, las cuales fueron implementadas en la generación de las mallas, que se definieron en los nodos y triángulos, para la modelación de las olas, corrientes y dinámicas de fluidos.

Aquí los archivos **.node** y **.ele**, se usan como la entrada para la construcción de dichas mallas, y simulan la dinámica de los fluidos.

CONSIDERACIONES TÉCNICAS

- *Precisión numérica*: Se utilizaron números de punto flotante con suficiente precisión para evitar errores en la triangulación y garantizar la estabilidad del algoritmo.
- *Optimización*: El algoritmo está optimizado para funcionar en entornos de bajo consumo de recursos.
- *Integración con herramientas*: Se asegura de que sean compatibles con MATLAB y SWAN.
- *Gestión de errores*: verifica la validez del .poly, y se maneja los errores de memoria y escritura/lectura de archivos.
- *Visualización*: validar gráficamente que la triangulación generada sea correcta antes de utilizarla en simulaciones.

ESTRUCTURA DE DATOS PRINCIPALES

1. *Punto*: almacena las coordenadas (x,y) de un punto en el plano.
    - Mantiene el índice original del punto en el conjunto de datos.

```
struct Punto {
    double x;
    double y;
    int indice;
}
```

2. *Triángulo*: almacena los tres vértices del triángulo.
    - Mantiene referencias a los triángulos vecinos.
    - Indica si es parte del super-triángulo inicial.

```
struct Triangulo {
    struct Punto *vertices[3];
    struct Triangulo *vecinos[3];
    int indices[3];
    int esTrianguloSuper;
}
```

FUNCIONES PRINCIPALES

1. *Lectura de datos*: lee puntos desde un archivo de formato .poly.
    - *Parámetros*:
        - nombreArchivo: ruta al archivo .poly.
        - numPuntos: puntero donde se almacena el número de puntos.
    - *Retorno*: arreglo de puntos leídos (NULL en caso de error).

```
struct Punto* leerArchivoPoly(const char* nombreArchivo, int* numPuntos)
```

2. *Triangulación*: realiza la triangulación de Delaunay completa.
    - *Algoritmo*:
        - Ordena los puntos por coordenada x.
        - Divide el conjunto de puntos recursivamente.
        - Combina las triangulaciones parciales.
    - *Complejidad*: O (n log n) en promedio.

```
struct Triangulacion* triangulacionDelaunay(struct Punto *puntos, int numPuntos)
```

3. *Verificación de Delaunay*: verifica el criterio del círculo vacío de Delaunay.
    - *Criterio*: Ningún punto debe de estar dentro del círculo circunscrito de cualquier ángulo.

```
int puntoEnCircunscrito(struct Triangulo *t, struct Punto *p)
```

FORMATO DE ARCHIVOS

1. *Archivo de entrada (.poly)*
    - Primera línea:
        - *<#vértices> <dimensión> <# atributos> <marcadores>*
    - Líneas siguientes:
        - *<índice> <x> <y> <marca>*
2. *Archivos de salida*
    - .node
        - Contiene las coordenadas de todos los puntos.
        - Formato: *<índice> <x> <y> <marca>*
    - .ele
        - Contiene la lista de triángulos.
        - Formato: *<índice_triángulo> <punto1> <punto2> <punto3>*

USO DEL PROGRAMA

1. Compilación:
    - *gcc delaunay3.c -o delaunay3.exe -lm*
2. Ejecución:
    - *./delaunay3.exe input.poly*