Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos
Taller 5: Grafos, 2024-10

1 Objetivo

Utilizar grafos para representar objetos (mallas) tridimensionales. En particular, se requiere aplicar algoritmos de búsqueda de rutas de costo mínimo para encontrar líneas de corte en mallas tridimensionales.

2 Recordatorio: compilación con g++

La compilación con g++ (compilador estándar que será usado en este curso para evaluar y calificar las entregas) se realiza con los siguientes pasos:

1. **Compilación**: de todo el código fuente compilable (**ÚNICAMENTE LOS ARCHIVOS CON EXTENSIONES** `*.c`, `*.cpp`, `*.cxx`)
   `g++ -std=c++11 -c *.c *.cxx *.cpp`
2. **Encadenamiento**: de todo el código de bajo nivel en el archivo ejecutable
   `g++ -std=c++11 -o nombre_de_mi_programa *.o`

   Nota: Estos dos pasos (compilación y encadenamiento) pueden abreviarse en un sólo comando:
   `g++ -std=c++11 -o nombre_de_mi_programa *.c *.cxx *.cpp`
3. **Ejecución**: del programa ejecutable anteriormente generado
   `./nombre_de_mi_programa`

ATENCIÓN: Los archivos de encabezados (*.h, *.hpp, *.hxx) **NO SE COMPILAN**, se incluyen en otros archivos (encabezados o código). Así mismo, los archivos de código fuente (*.c, *.cpp, *.cxx) **NO SE INCLUYEN**, se compilan. Si el programa entregado como respuesta a este Taller no atiende estas recomendaciones, automáticamente se calificará la entrega sobre un 25% menos de la calificación máxima.

3 Descripción del problema

Una malla es un objeto que se utiliza para representar superficies en 3D. Una malla se puede ver como un par ordenado $\mathcal{M} = [\mathcal{P}, \mathcal{E}]$, donde $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}$ es un conjunto de puntos en 3D y $\mathcal{E} = \{< i, j >; \mathbf{p}_i \neq \mathbf{p}_j \wedge \mathbf{p}_i, \mathbf{p}_j \in \mathcal{P}\}$ es el conjunto de aristas de la malla.

Uno de los usos de este tipo de objetos es el "mapeo" de texturas. "Mapear" una textura consiste en asignar a cada punto de la malla un color específico para lograr un efecto visual adecuado (ver Figura 1).

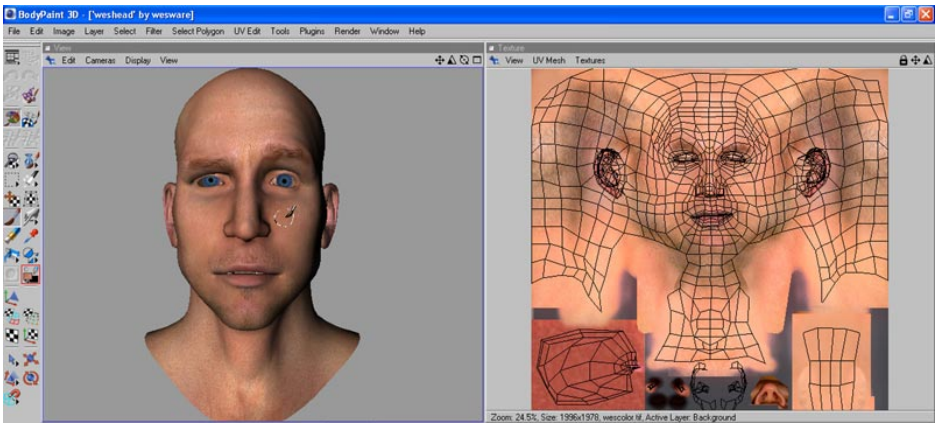

Figure 1: Mapeado de texturas. Tomado de `http://www.cosc.brocku.ca/Offerings/3P98/course/lectures/texture`

Note que la malla (izquierda) tiene la geometría del objeto, mientras que la textura (derecha) está "aplanada". Para facilitar el trabajo de los coloristas, se debe "aplanar" la malla. El "aplanado" se realiza siguiendo una metáfora de "tijeras": hay que encontrar la línea por donde la malla se va a cortar.

Una de las líneas posibles de corte es precisamente la línea que conecta los dos puntos más alejados de la malla, pero manteniendo esa distancia óptima (es decir, que la distancia euclideana entre puntos vecinos sea la menor posible). La Figura 2 muestra un ejemplo de una línea de corte (en color anaranjado) para una malla con forma elipsoide. Note que la línea de corte conecta los puntos más alejados de la malla y la longitud total de la línea de corte es la más corta posible entre dichos puntos.

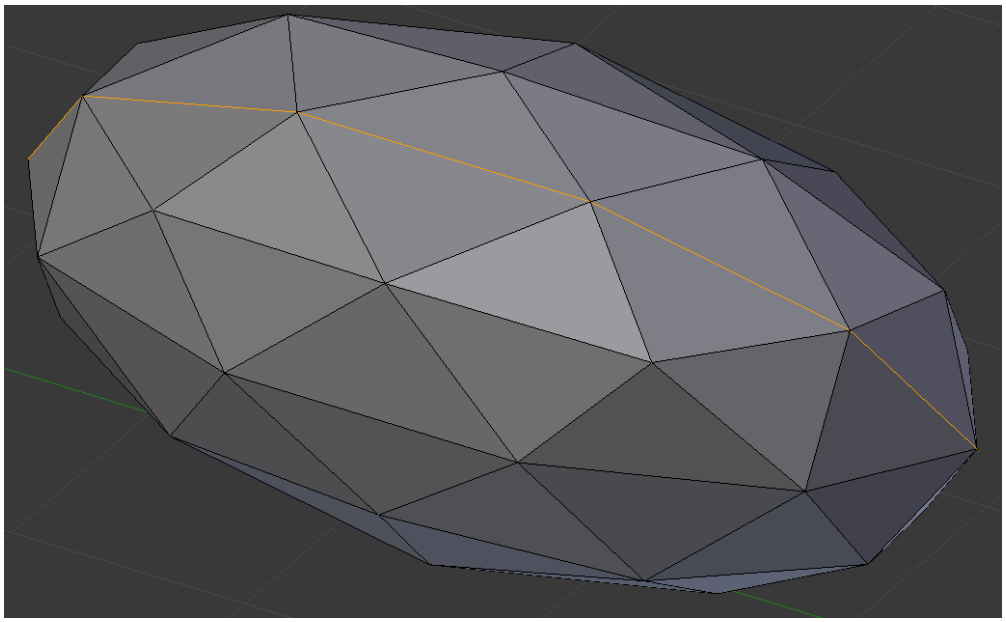

Figure 2: Línea de corte de una malla elipsoide

Se requiere entonces desarrollar un programa en C++ para calcular esta línea, a partir de una malla dada por el usuario. La malla se lee de un archivo de texto con el siguiente formato:

N
$x_0 \quad y_0 \quad z_0$
$x_1 \quad y_1 \quad z_1$
...
$x_{N-1} \quad y_{N-1} \quad z_{N-1}$
E
$s_0 \quad e_0$
$s_1 \quad e_1$
...
$s_{E-1} \quad e_{E-1}$

Donde:

- N es el número de puntos de la malla.
- $x_i \quad y_i \quad z_i$ es la tupla con las coordenadas del i-ésimo punto.
- E es número de aristas de la malla.
- $s_j \quad e_j$ es la dupla con los índices de los puntos que conforman la j-ésima arista. Por ejemplo, una línea con los valores

3 5

indica que esta arista conecta el punto de coordenadas $x_3 \ y_3 \ z_3$ con el punto de coordenadas $x_5 \ y_5 \ z_5$.

El programa debe mostrar por pantalla la secuencia de puntos 3D que conforman la línea de corte.

4 Descripción de la implementación

Para resolver el problema de línea de corte, se propone una implementación contenida en el archivo fuente "`taller_5_grafos.cxx`". El programa recibe tres parámetros por la línea de comandos:

- el nombre del archivo que contiene la malla,
- el índice del punto desde donde empieza la línea de corte y,
- el índice del punto hasta donde llega la línea de corte.

Para generar un archivo con el camino generado, se puede redirigir la salida estándar a un archivo de texto, de la siguiente forma:

```
./ejecutable_programa archivo_malla.txt indice_origen indice_destino > archivo_salida.txt
```

5 Desarrollo del taller

El desarrollo del taller consistirá en diseñar (utilizando la plantilla de diseño de TADs vista en clase) e integrar el código necesario para que el archivo fuente "`taller_5_grafos.cxx`" implemente un programa que imprima por pantalla la línea de corte para una malla dada. Para completar el taller, deben realizarse las siguientes actividades:

1. Estudiar el archivo de código fuente entregado. Es importante entender completamente los procesos allí descritos. Si alguna porción de código no es clara o no se entiende, pregunte al profesor o al monitor de la clase.
2. Reemplazar las secciones de código marcadas como "`TODO`" con las porciones de código adecuadas para el correcto funcionamiento del programa que se pide.
   - `TODO 1`: definir el tipo de dato del grafo a utilizar, que soporte un tipo de dato `Point` como vértice y un valor real como costo de la conexión.
   - `TODO 2`: declarar el grafo.
   - `TODO 3`: agregar cada punto de la malla como vértice del grafo.
   - `TODO 4`: calcular el costo de la conexión (usando la distancia euclidiana entre puntos) y agregar la arista correspondiente en el grafo no dirigido (ambas direcciones de la conexión).
   - `TODO 5`: utilizar un algoritmo de búsqueda de rutas de costo mínimo (como Dijkstra o Prim, por ejemplo), para identificar los índices de los vértices que hacen parte de la ruta de costo mínimo, imprimiendo en pantalla la cantidad de vértices y las coordenadas de cada uno, en el orden que permita reconstruir la ruta.

6 Evaluación

La entrega se hará a través de la correspondiente asignación de BrightSpace, antes de la medianoche del martes 21 de mayo. Se debe entregar un único archivo comprimido (único formato aceptado: .zip), nombrado con los apellidos de los integrantes del grupo. Este comprimido debe contener, dentro de un mismo directorio (sin estructura de carpetas interna), el documento de diseño (.pdf) y el código fuente modificado y funcional (.h, .hxx, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

La evaluación del taller tendrá la siguiente escala para el código a completar:

- **Excelente** (**5.0/5.0**): El estudiante diseñó correctamente (siguiendo la plantilla) e implementó una solución que compila, ejecuta y genera resultados correctos.
- **Bueno** (**3.0/5.0**): El estudiante diseñó correctamente (siguiendo la plantilla) e implementó una solución parcialmente correcta (el código compila, ejecuta, pero no genera resultados correctos).
- **No fue un trabajo formal de ingeniería** (**2.5/5.0**): El estudiante implementó una solución completa o parcial, y/o no la diseñó correcta o completamente.
- **Necesita mejoras sustanciales** (**2.0/5.0**): El estudiante diseñó y/o implementó una solución, pero genera errores de ejecución (violaciones de segmento, por ejemplo).
- **Malo** (**1.0/5.0**): El código entregado por el estudiante no compila en el compilador g++ (mínimo versión número 4.5).
- **No entregó** (**0.0/5.0**).