Relatório de Implementação — Estrutura DCEL e Validação Topológica

Efi Fuchs Santos da Silva - 20211775
UFPR - 29.05.2025

Introdução do Trabalho

Este trabalho teve como objetivo implementar um sistema para validar e construir uma subdivisão planar a partir de entradas que definem vértices e faces. A ideia foi montar uma estrutura chamada **DCEL (Doubly Connected Edge List)** para representar graficamente essas subdivisões e garantir que estejam bem definidas topologicamente.

Etapas da Implementação

A primeira etapa foi ler os vértices da entrada e armazená-los em uma estrutura `Point`, sendo que cada vértice é identificado pelas suas coordenadas (x, y). Para isso, criei um vetor de pontos e uma função auxiliar `find_or_create_vertex` que evita criar vértices duplicados e garante que todos os objetos `Vertex` sejam reutilizáveis.

Depois disso, foram lidas as faces. Cada face é formada por um conjunto de segmentos (arestas), e esses segmentos foram armazenados em uma estrutura `Face`, usando pares de pontos consecutivos. Também foi feito o fechamento do polígono (ligando o último ponto ao primeiro).

Validação da Topologia

A validação foi uma parte fundamental e aconteceu em três etapas:

1. **Checagem de duplicação de arestas**: Verifiquei se existia algum segmento repetido na mesma direção usando uma hash table (`EdgeMap`). Caso fosse encontrado, a subdivisão era considerada inválida (`"não subdivisão planar"`).

2. **Checagem de arestas abertas**: Aqui, validei se para cada aresta de uma face existia outra aresta com direção contrária em uma face vizinha. Se uma aresta não tiver par, a subdivisão é considerada `"aberta"`.

3. **Checagem de interseção indevida**: Foram checadas interseções entre segmentos que não fossem apenas nos vértices. Se duas arestas de faces diferentes se cruzarem fora dos vértices, a subdivisão é considerada `"superposta"`.

Construção da DCEL

Após a validação, se a topologia estiver correta, a estrutura **DCEL** é construída. Para isso:

- Cada `HalfEdge` (semi-aresta) é criada com um identificador único.
- Para cada segmento, são criadas duas semi-arestas gêmeas (uma para cada face).
- As ligações `next`, `prev` e `twin` são feitas para garantir a navegação pelas bordas da face.
- Cada vértice guarda uma semi-aresta incidente, e cada face aponta para uma semi-aresta da sua borda externa.

Por fim, foi feito um `printf` de todos os vértices com as informações associadas e a listagem das semi-arestas, seguindo o formato exigido.

Algoritmos Utilizados

- **Hashing personalizado para segmentos**: Foram utilizados hashcodes baseados nas coordenadas dos pontos para evitar colisões e permitir acesso eficiente.

- **Verificação de interseção geométrica**: Foi usado o teste de orientação (produto vetorial) para verificar se dois segmentos se cruzam.

- **Construção de DCEL**: Baseado em estruturas clássicas de grafos planares, com criação explícita de gêmeas e ponteiros de navegação.