Implementação de DCEL para Subdivisão Planar

Trabalho de Geometria Computacional

Richard Fernando Heise Ferreira
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

26 de maio de 2025

Resumo

Este trabalho apresenta a implementação de uma Doubly Connected Edge List (DCEL) para representação de subdivisões planares. O sistema desenvolvido é capaz de ler uma descrição de malha planar, verificar sua validade topológica e gerar a estrutura DCEL correspondente. Foram implementados algoritmos para detecção de malhas abertas, não-planares e com faces sobrepostas, seguindo os princípios fundamentais da geometria computacional.

1 Introdução

A representação eficiente de subdivisões planares é essencial para diversas aplicações em computação gráfica, sistemas de informação geográfica e modelagem geométrica. Este trabalho foca na implementação da estrutura DCEL, que permite:

- Representação compacta de vértices, arestas e faces
- Navegação eficiente entre elementos adjacentes
- Verificação de propriedades topológicas
- Suporte a operações geométricas complexas

2 Algoritmos Implementados

2.1 Malha Aberta e Não-Subdivisão Planar

Verifica-se se a malha é topologicamente fechada e representa uma subdivisão planar válida. Para isso, todas as arestas direcionadas são mapeadas às respectivas faces e é verificada a existência da aresta gêmea. Garante-se que cada par de arestas (original e inversa) ocorra exatamente duas vezes. Caso uma aresta ocorra menos ou mais de duas vezes, a malha é considerada aberta ou inválida como subdivisão do plano, respectivamente.

2.2 Superposição

Detecta-se auto-interseções e posicionamentos inválidos de vértices nas faces. Inicialmente, testa-se se arestas não adjacentes de uma mesma face se cruzam geometricamente, com base em orientações vetoriais e casos especiais de colinearidade. Em seguida, para cada face orientada no sentido anti-horário, verifica-se se algum vértice externo a ela encontra-se sobre sua borda ou em seu interior. A verificação utiliza colinearidade para detectar presença na borda e a técnica de *ray casting* para determinar inclusão interna. Qualquer detecção de sobreposição ou ponto mal posicionado invalida a malha.

2.3 Construção da DCEL

A construção da estrutura DCEL segue três etapas principais:

1. Criação dos vértices a partir das coordenadas de entrada, na mesma ordem.
2. Construção das semi-arestas e conexão das arestas gêmeas.
3. Estabelecimento das relações de adjacência (next/prev).

3 Estrutura do Código

O projeto está organizado nos seguintes componentes:

3.1 Classes Principais

- `Vertex`: Armazena coordenadas e ponteiro para aresta incidente
- `HalfEdge`: Representa semi-arestas com referências para origem, face, next, prev e twin
- `Face`: Contém ponteiro para componente externo.
- `DCEL`: Estrutura principal que agrega todos os elementos

3.2 Funções Principais

- `buildFromMesh`: Constrói a DCEL a partir dos dados de entrada, na mesma ordem.
- `createHalfEdgesAndFaces`: Implementa o algoritmo de construção das arestas e faces.
- `printDCELOutput`: Gera a saída no formato especificado.
- `ValidateEdges`: Realiza as verificações topológicas.

4 Testes e Validação

O sistema foi validado com diversos casos de teste, incluindo:

- Malhas válidas com diferentes complexidades
- Casos de malhas abertas

- Subdivisões não-planares
- Faces com auto-interseção

Os testes podem ser encontrados na pasta *tests* onde também há um script em python para geração da representação visual de cada teste. Na pasta *inputs* os testes são armazenados, já na pasta *outputs* é onde se espera a saída de cada teste (caso o programa seja rodado a partir da raiz com o script *run_tests.sh* isso é feito automaticamente). Já na pasta *validator* há um script em python que valida as respostas geradas e transforma em imagem a DCEL válida.