Technische Universität Braunschweig

Einführung in algorithmische Geometrie – Overlays

Arne Schmidt

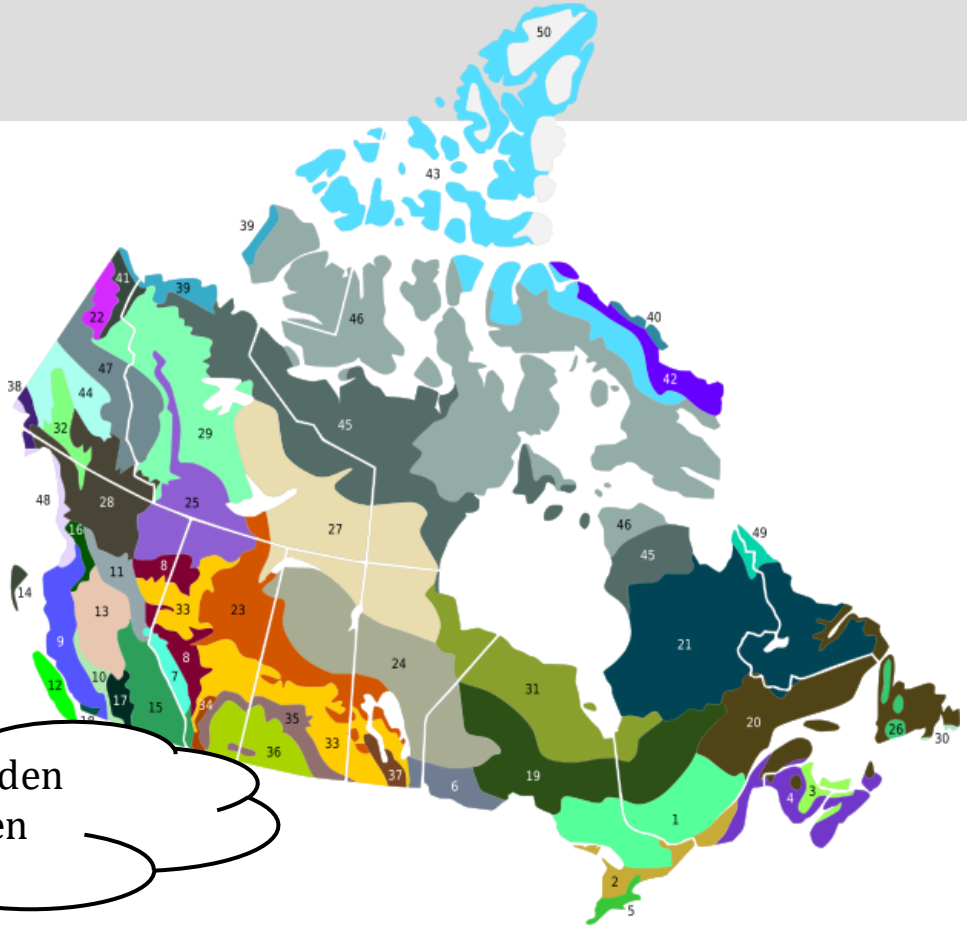
Ökoregionen in Kanada

Eine lose Sammlung an Linien ist in der Praxis selten

Es ist viel mehr eine Sammlung von Flächen.

Wie können wir das effizient speichern?

Wie ist das mit Überlappenden regionen? (bspw. Regionen und Provinzen)

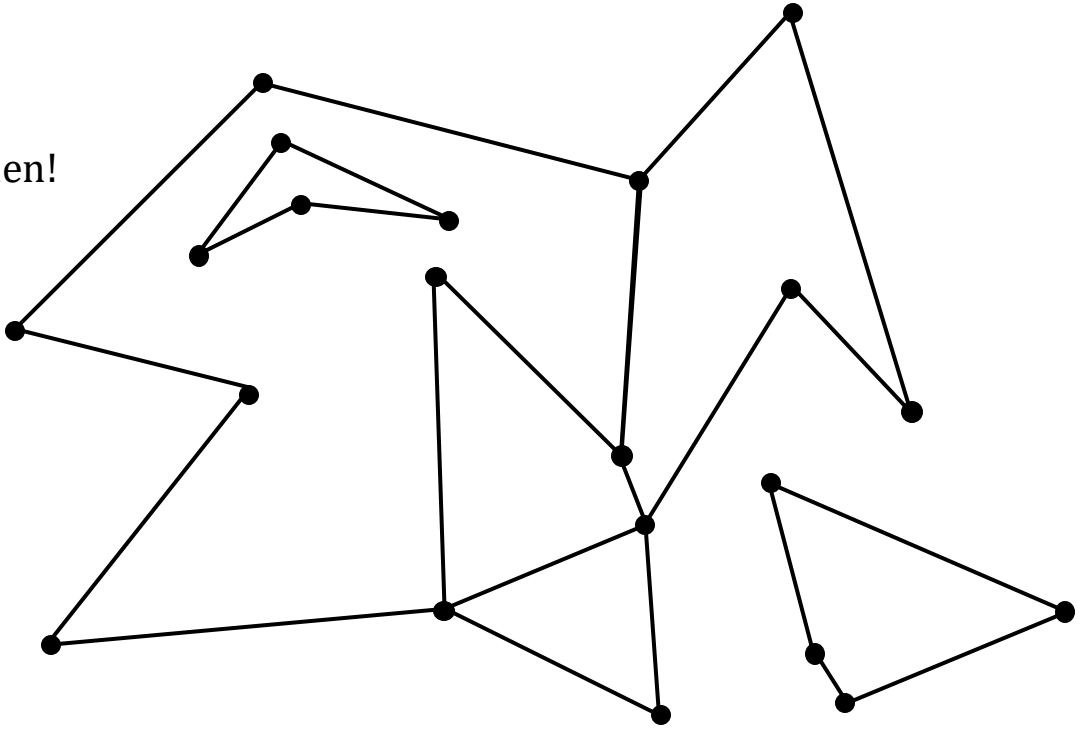


Regionen vereinfacht dargestellt

Eine Art Graph?
- Zu abstrakt
- Keine geometrischen Informationen!

Was haben wir?
- Begrenzte Flächen
- Flächen, die Regionen enthalten.

Wie können wir beispielsweise…?
- Ränder ablaufen
- Regionen ablaufen
- Benachbarte Regionen ausgeben
- …

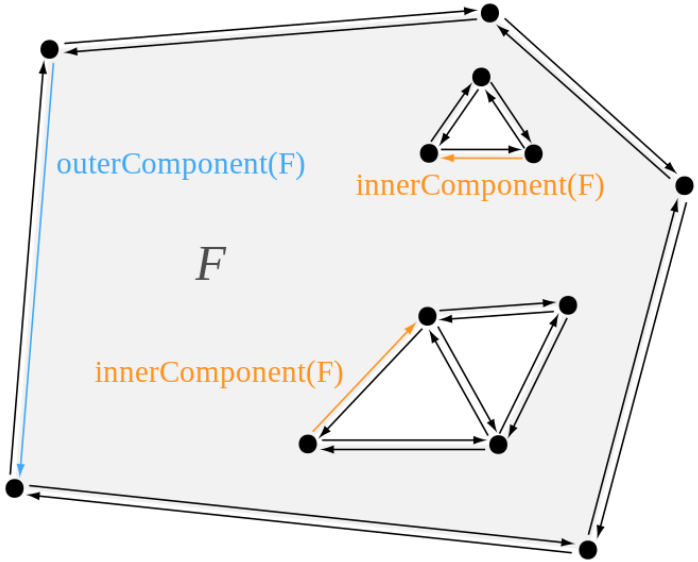


Datenstruktur - Bedingungen

Wünsche an die Datenstruktur:

- Man kann sich von jedem Teil des Arrangements zu jedem anderen Teil des Arrangements „bewegen“.
- Ich benötige Pointer von jeder Fläche zu mind. einer seiner begrenzenden Kante.
- Ich muss von Kante zu Kante springen können.
- Für Kanten muss ich inzidente Knoten kennen.
- An Knoten muss ich inzidente Kanten kennen.

Kapitel 2.2.3 – DCEL



https://de.wikipedia.org/wiki/Half-Edge-Datenstruktur#/media/Datei:Dcel-components.svg

Datenstruktur - DCEL

Definition 2.23a (DCEL):
In einer Doubly-Connected Edge-List (DCEL) für ein Arrangement werden folgende Informationen inkl. Attributen gespeichert:

- Halbkante (engl. **Half-Edges**, gerichtete Kante zwischen zwei Knoten):
  - Pointer zur nachfolgenden Half-Edge (**next**)
  - Pointer zur vorhergehenden Half-Edge (**prev**)
  - Pointer zur entgegengesetzten Half-Edge (**twin**)
  - Pointer zum repräsentierten Face (**IncidentFace**)
  - Pointer zum Startknoten (**Origin**)
- Knoten (engl. **Vertex**, Start/Ende von Kanten):
  - Koordinaten
  - Pointer auf eine ausgehende Half-Edge (**IncidentEdge**)
- Fläche (engl. **Face**):
  - Pointer auf eine äußere Half-Edge (**OuterComponent**)
  - Liste von Pointern auf eine innere Half-Edge pro Komponente (**InnerComponent**)

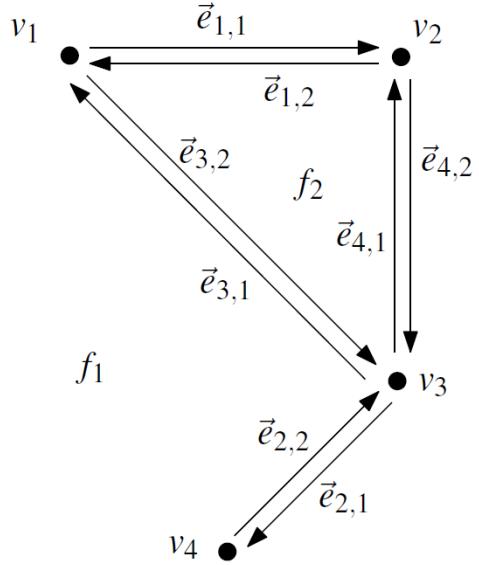
Datenstruktur - DCEL

Definition 2.23b (DCEL):
In einer Doubly-Connected Edge-List (DCEL) für ein Arrangement werden folgende Informationen inkl. Attributen gespeichert:

- Fläche (engl. **Face**):
  - Pointer auf eine äußere Half-Edge (**OuterComponent**)
  - Liste von Pointern auf eine innere Half-Edge pro Komponente (**InnerComponent**)

Die Half-Edges, die eine Fläche begrenzen sind dabei gegen den Uhrzeigersinn orientiert. Für innere Komponenten sind diese mit dem Uhrzeigersinn orientiert.

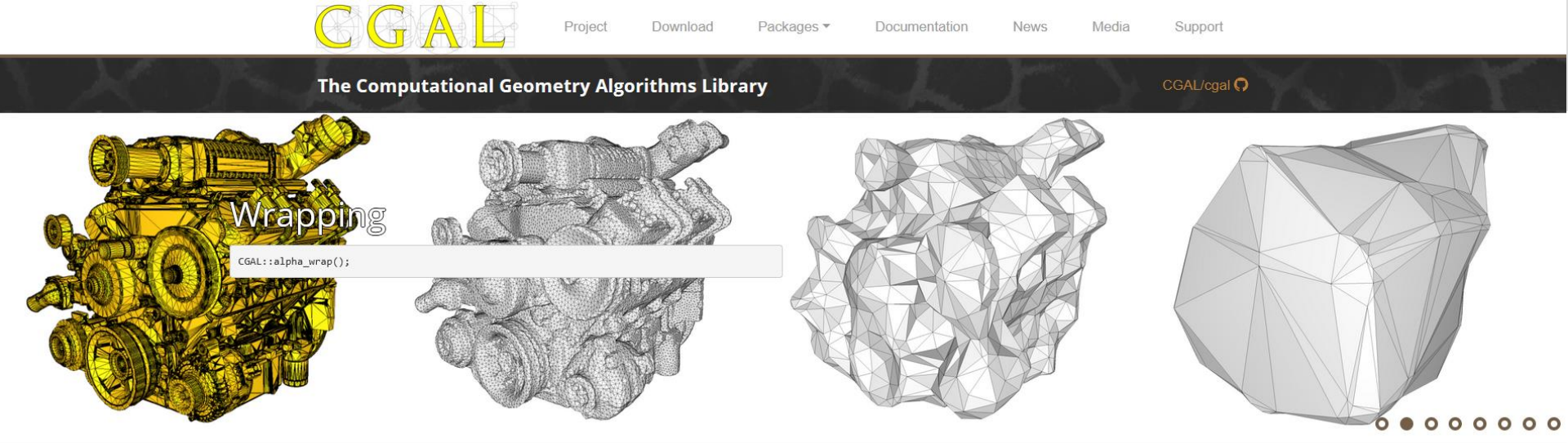
DCEL Beispiel



Vertex	Coordinates	IncidentEdge
v_1	$(0,4)$	$\vec{e}_{1,1}$
v_2	$(2,4)$	$\vec{e}_{4,2}$
v_3	$(2,2)$	$\vec{e}_{2,1}$
v_4	$(1,1)$	$\vec{e}_{2,2}$

Face	OuterComponent	InnerComponents
f_1	**nil**	$\vec{e}_{1,1}$
f_2	$\vec{e}_{4,1}$	**nil**

Half-edge	Origin	Twin	IncidentFace	Next	Prev
$\vec{e}_{1,1}$	v_1	$\vec{e}_{1,2}$	f_1	$\vec{e}_{4,2}$	$\vec{e}_{3,1}$
$\vec{e}_{1,2}$	v_2	$\vec{e}_{1,1}$	f_2	$\vec{e}_{3,2}$	$\vec{e}_{4,1}$
$\vec{e}_{2,1}$	v_3	$\vec{e}_{2,2}$	f_1	$\vec{e}_{2,2}$	$\vec{e}_{4,2}$
$\vec{e}_{2,2}$	v_4	$\vec{e}_{2,1}$	f_1	$\vec{e}_{3,1}$	$\vec{e}_{2,1}$
$\vec{e}_{3,1}$	v_3	$\vec{e}_{3,2}$	f_1	$\vec{e}_{1,1}$	$\vec{e}_{2,2}$
$\vec{e}_{3,2}$	v_1	$\vec{e}_{3,1}$	f_2	$\vec{e}_{4,1}$	$\vec{e}_{1,2}$
$\vec{e}_{4,1}$	v_3	$\vec{e}_{4,2}$	f_2	$\vec{e}_{1,2}$	$\vec{e}_{3,2}$
$\vec{e}_{4,2}$	v_2	$\vec{e}_{4,1}$	f_1	$\vec{e}_{2,1}$	$\vec{e}_{1,1}$



CGAL is an open source software project that provides easy access to efficient and reliable geometric algorithms in the form of a C++ library. CGAL is used in various areas needing geometric computation, such as geographic information systems, computer aided design, molecular biology, medical imaging, computer graphics, and robotics.

The library offers data structures and algorithms like triangulations, Voronoi diagrams, Boolean operations on polygons and polyhedra, point set processing, arrangements of curves, surface and volume mesh generation, geometry processing, alpha shapes, convex hull algorithms, shape reconstruction, AABB and KD trees... Explore the complete list of features and capabilities by visiting the CGAL Package Overview.

The CGAL data structures and algorithms are distributed under a dual license, namely under the GPL v3+ and, alternatively, under a commercial license by GeometryFactory.

Latest News

- December 2024 Tetrahedral Mesh Generation Improvements -- Tetrahedral Mesh Generation Improvements in CGAL 6
- October 2024 New CGAL versions: 5.5.5, 5.6.2, 6.0, and 6.0.1
- July 2024 New in CGAL: Basic Viewer

Latest Releases

- **Latest stable release:** CGAL 6.0.1

Previous Releases

- October 2024 CGAL 5.6.2

CGAL 6.0.1 - 2D Arrangements

Classes

DCEL Concepts

2D Arrangement Reference » Concepts

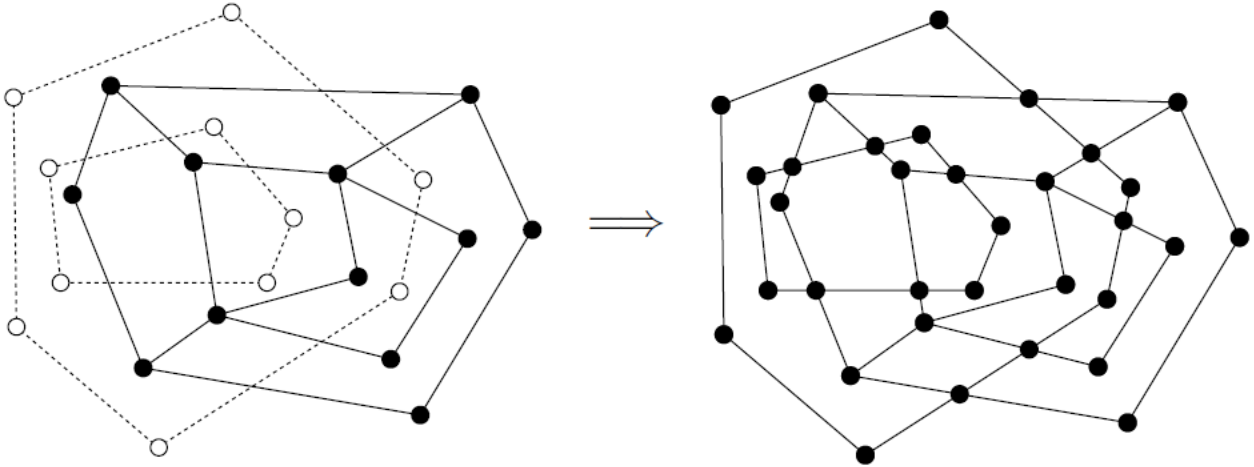
Concepts

concept ArrangementDcel

A doubly-connected edge-list (DCEL for short) data-structure. It consists of three containers of records: vertices V, halfedges E, and faces F. It maintains the incidence relation among them. The halfedges are ordered in pairs sometimes referred to as twins, such that each halfedge pair represent an edge. More...

- July 2024 New in CGAL: Basic Viewer
- October 2024 CGAL 5.6.2

Kapitel 2.2.4 – Overlays



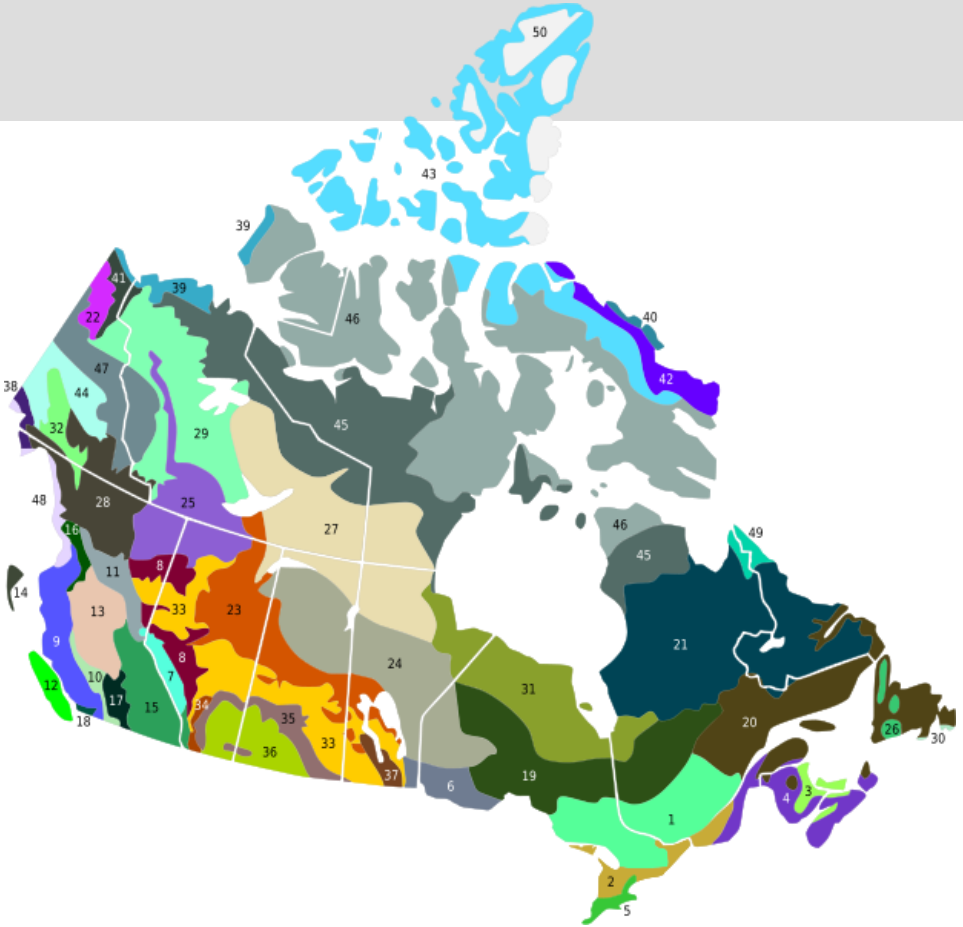
De Berg et al.: Computational Geometry – Algorithms and Applications

Overlay

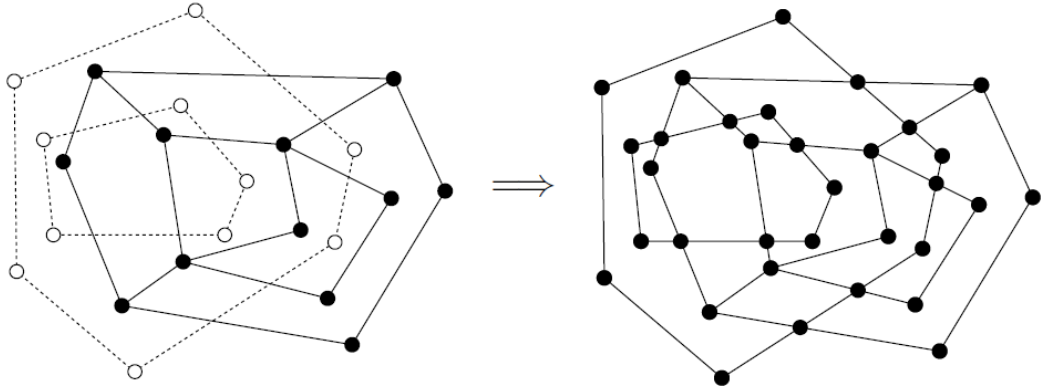
Im Fall von Kanada möchte man nicht nur wissen, in welcher Ökoregion man ist, sondern auch in welcher Provinz.

Man kann für beide Arrangements separat eine Lokalisierungsanfrage durchführen.

Aber was passiert, wenn wir k viele Arrangements besitzen?
→ Können wir alle Arrangements zusammenlegen?



Ideen für einen Algorithmus



Das Ergebnis sieht ähnlich zu dem Intersection-Problem aus. Können wir den Sweep-Line-Algorithmus wiederverwenden?

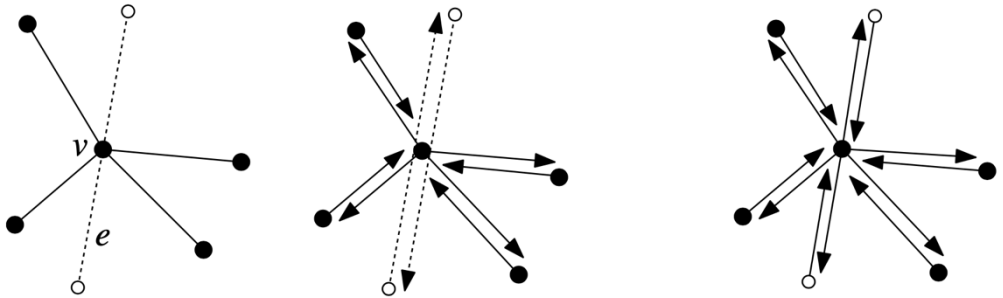
Wichtig:

- Jeder vorhandene Knoten bleibt bestehen.
- Half-Edges spalten sich auf und erzeugen neue Knoten und Half-Edges. Nur bei einer Intersection!
- Bestehende Faces werden weiter unterteilt. Wie erkennt man das?

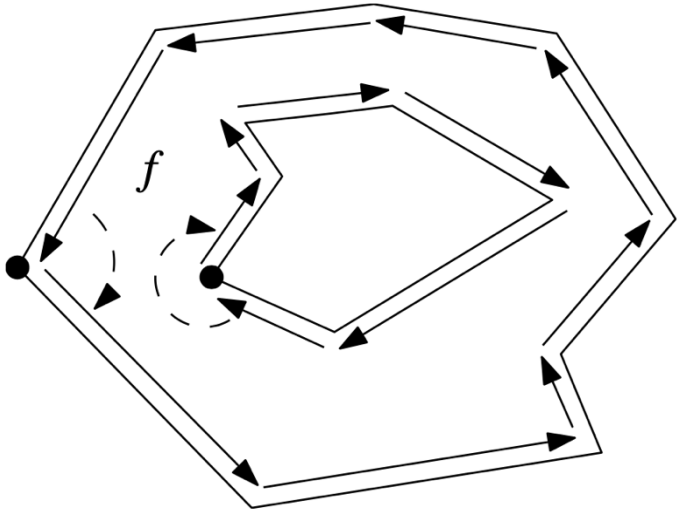
Schauen wir uns das an der Tafel an!

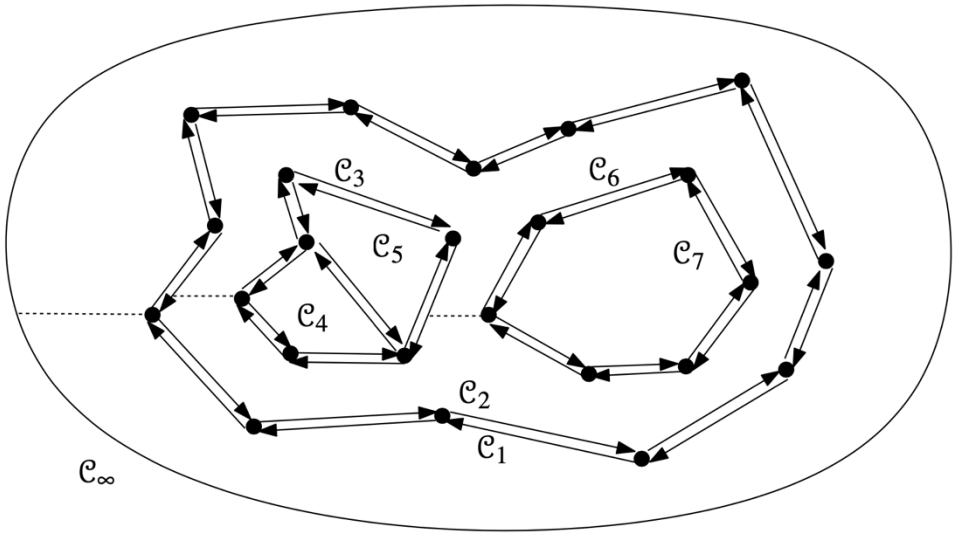
Theoreme

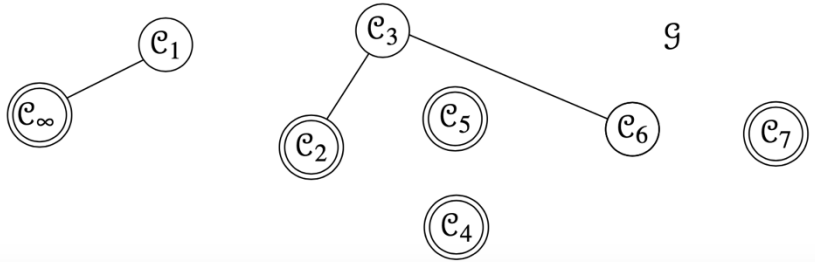
Theorem 2.24:
Seien $\mathcal{S}_1, \mathcal{S}_2$ zwei Arrangements mit Komplexität n_1 bzw. n_2 und sei $n = n_1 + n_2$. Das Overlay von $\mathcal{S}_1$ und $\mathcal{S}_2$ kann in Zeit $O(n \log n + k \log n)$ erzeugt werden, wobei k die Komplexität des Overlays ist.



Bestimmen der Faces







Theoreme

Theorem 2.24:
Seien $\mathcal{S}_1, \mathcal{S}_2$ zwei Arrangements mit Komplexität n_1 bzw. n_2 und sei $n = n_1 + n_2$. Das Overlay von $\mathcal{S}_1$ und $\mathcal{S}_2$ kann in Zeit $O(n \log n + k \log n)$ erzeugt werden, wobei k die Komplexität des Overlays ist.

Korollar 2.25:
Seien P_1, P_2 zwei Polygone mit n_1 bzw. n_2 Knoten und sei $n = n_1 + n_2$. Folgende Operationen können in Zeit $O(n \log n + k \log n)$ durchgeführt werden, wobei k die Komplexität der Ausgabe ist:

- Schnitt $P_1 \cap P_2$
- Vereinigung $P_1 \cup P_2$
- Differenz $P_1 \setminus P_2$