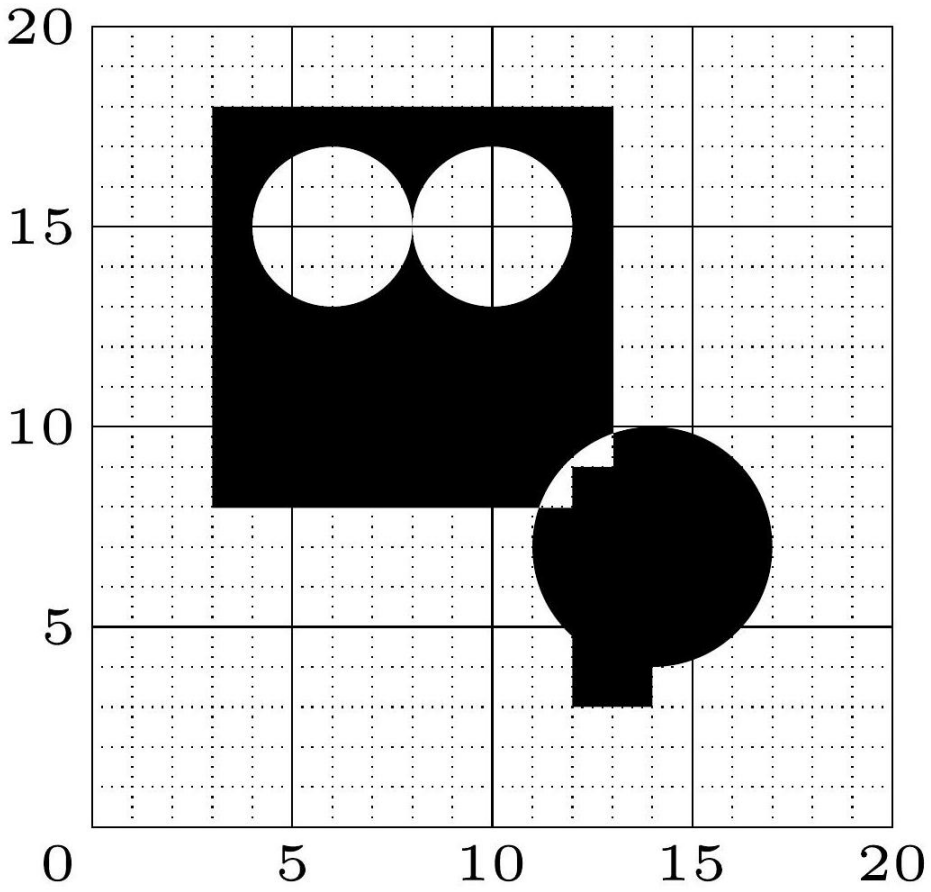
Assignment 11

Basic Techniques in Computer Graphics
WS 2022/2023
January 24, 2023

Frederik Muthers	412831
Tobias Broeckmann	378764
Debabrata Ghosh	441275
Martin Gäbele	380434

Exercise 1 Constructive Solid Geometry



(a) We can use the following abstract specification:

- A rectangle of size 10×10 (A square of size 10) with its lower left corner at $(3, 8)^T$
- A circle of radius 2 with center at $(6, 15)^T$
- A circle of radius 2 with center at $(10, 15)^T$
- A circle of radius 3 with center at $(14, 7)^T$
- A rectangle of size 2×6 with its lower left corner at $(12, 3)^T$

(b) For each shape described in the previous subtask, we can define an implicit function $p_i : \mathbb{R}^2 \to \mathbb{R}, \quad 1 \leq i \leq 5$ such that its value is negative on the inside and positive on the outside of the respective shape as follows:

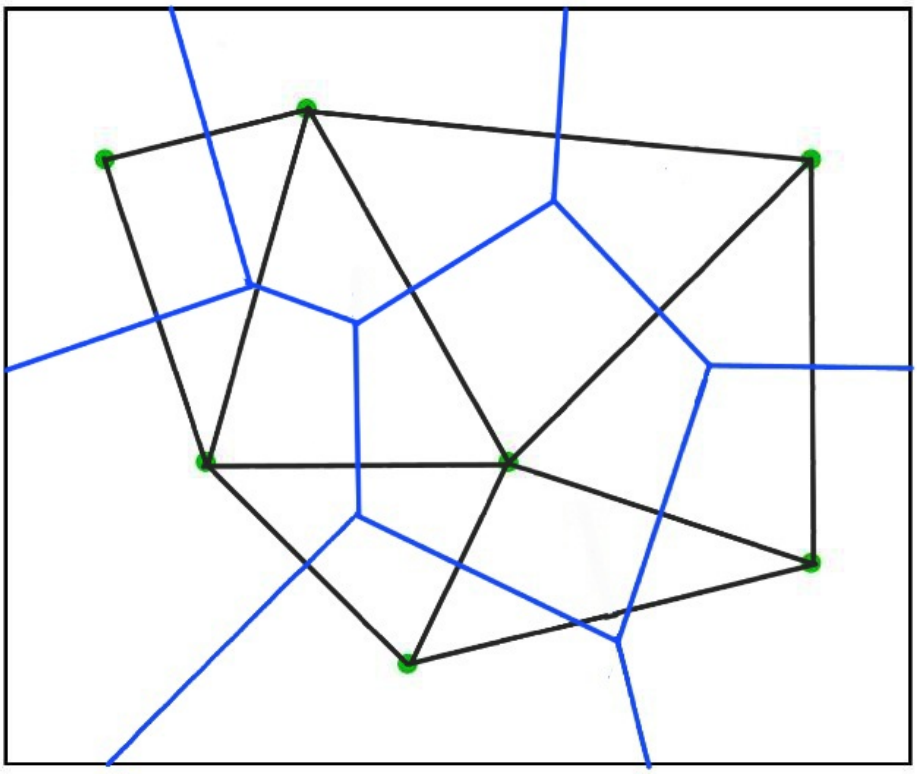
- $p_1(x, y) = max(3 - x, x - 13, 8 - y, y - 18)$
- $p_2(x, y) = (x - 6) \cdot (x - 6) + (y - 15) \cdot (y - 15) - 4$

- $p_3(x, y) = (x - 10) \cdot (x - 10) + (y - 15) \cdot (y - 15) - 4$
- $p_4(x, y) = (x - 14) \cdot (x - 14) + (y - 7) \cdot (y - 7) - 9$
- $p_5(x, y) = max(12 - x, x - 14, 3 - y, y - 9)$

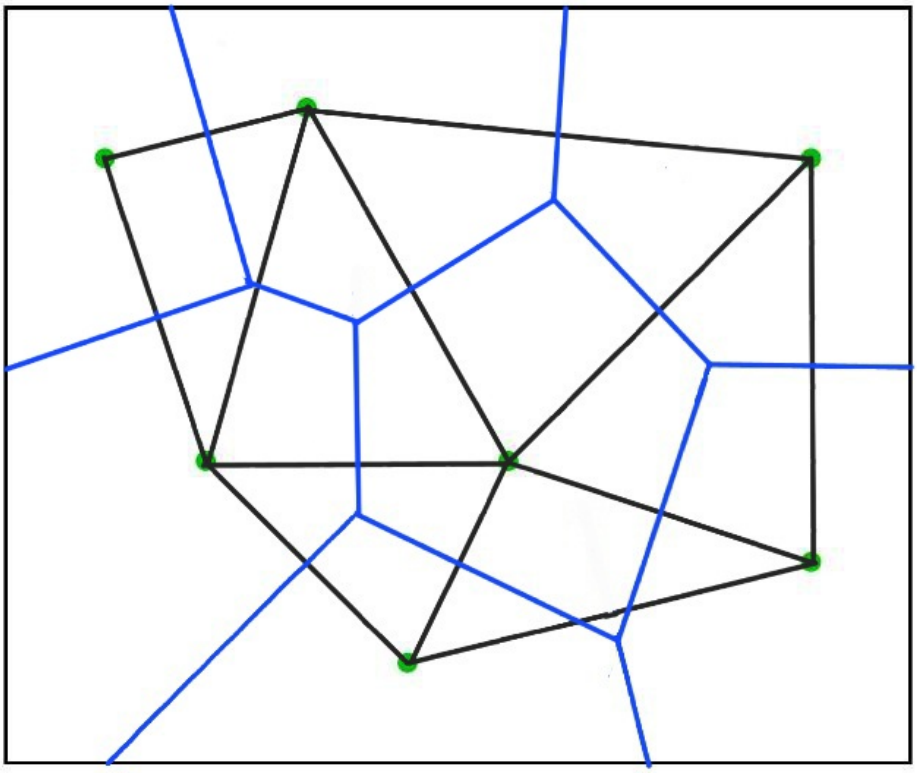
(c) We can combine p_1, p_4, p_5 as $p_{145} = (p_1 \oplus p_4) \cup p_5 = (p_1 \cdot p_4) \cup p_5 = min(p_1 \cdot p_4, p_5)$. Now we can combine p_{145}, p_2, p_3 as $p_{12345} = p_{145} - p_2 - p_3 = max(p_{145}, -p_2, -p_3) = max(min(p_1 \cdot p_4, p_5), -p_2, -p_3)$.

Exercise 2 Voronoi Diagram and Delaunay Triangulation

(a) Voronoi Diagram of the given point set with the boundaries shown by the blue lines:

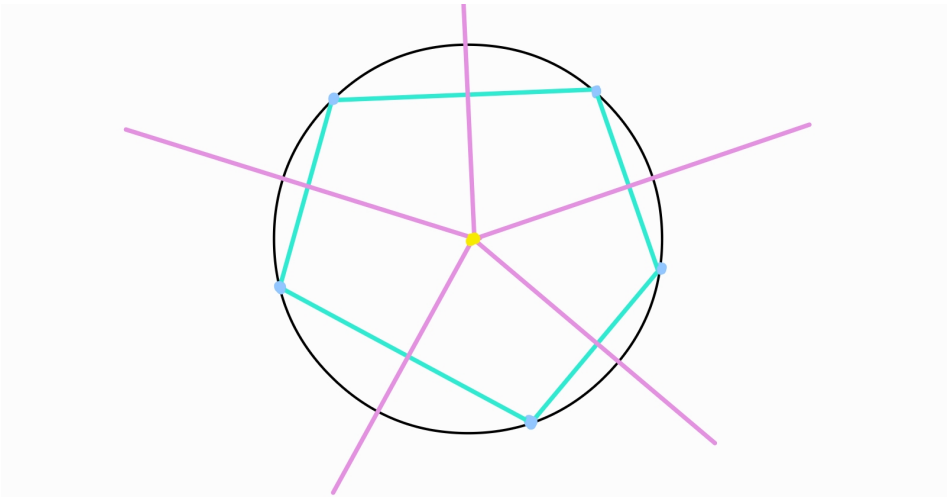


(b) Delaunay Triangulation of the given point set with the triangle edges shown by black lines:

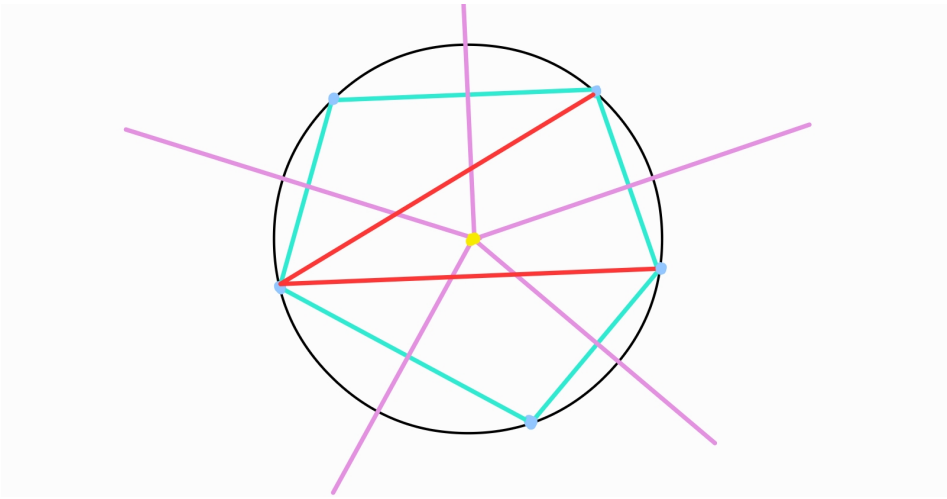


Exercise 3 Dual of Voronoi Diagrams

(a) In most cases, dualizing the Voronoi Diagram of a set of 2D points immediately yields a Delaunay Triangulation of the given points. However, there are exceptional cases where the dual is not a triangle mesh e.g. four points in a square configuration have a Voronoi Diagram that dualizes to a quadrilateral instead of two triangles. The reason being all four points in a square (vertices of the square) are concyclic. Since all those points lie on the same circle, the center of the circle is the point with the smallest distance where all the half-spaces meet. Hence such a dual structure. Similarly, we can find for any $n \geq 3, n$ such points in the same circle, resulting in a dual polygon with n sides. We can also argue in backwards direction as dual of a n-sided polygon is a Voronoi vertex with valance n which is a point with equal and the smallest distance to those n points and so those n points lie on a circle. For n = 5, following is an example sketch of the input point set, its Voronoi Diagram and the resulting dual mesh:

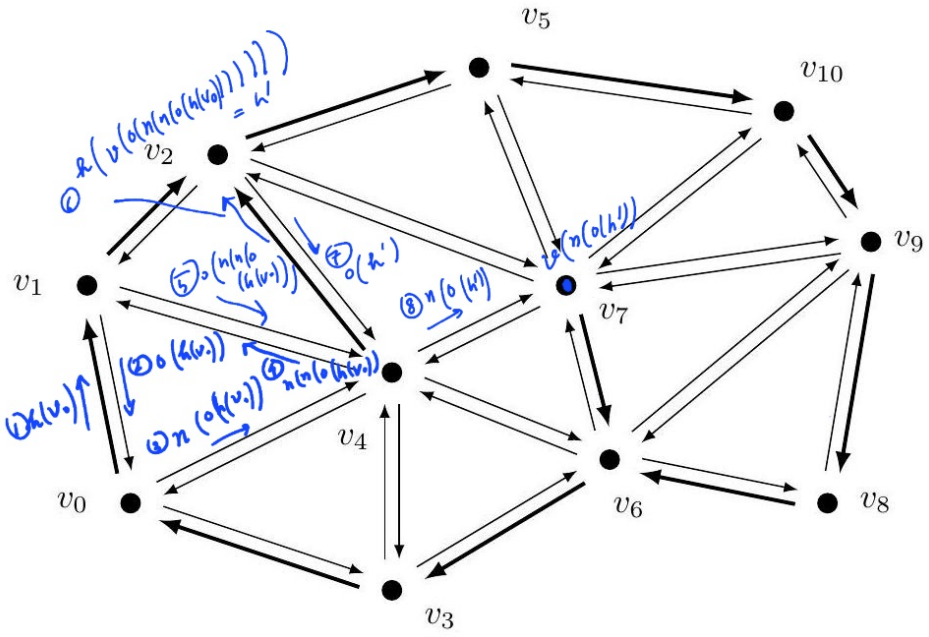


(b) Given a point set constructed accordingly to task (a) and the resulting n-sided polygon (with $n > 3$), the resulting polygon is triangulated by arbitrarily picking one vertex v and connecting it to all other vertices that aren't connected to v yet. The resulting mesh will always be a Delaunay triangulation. Since for every triangle, the circumcircle is the same as the original circle used in the construction, and since all points lie on this circle only, it is an empty circle by construction and the Delaunay triangulation is achieved. We can see an example for the case we drew in the task (a):



Exercise 4 Halfedge Data Structures

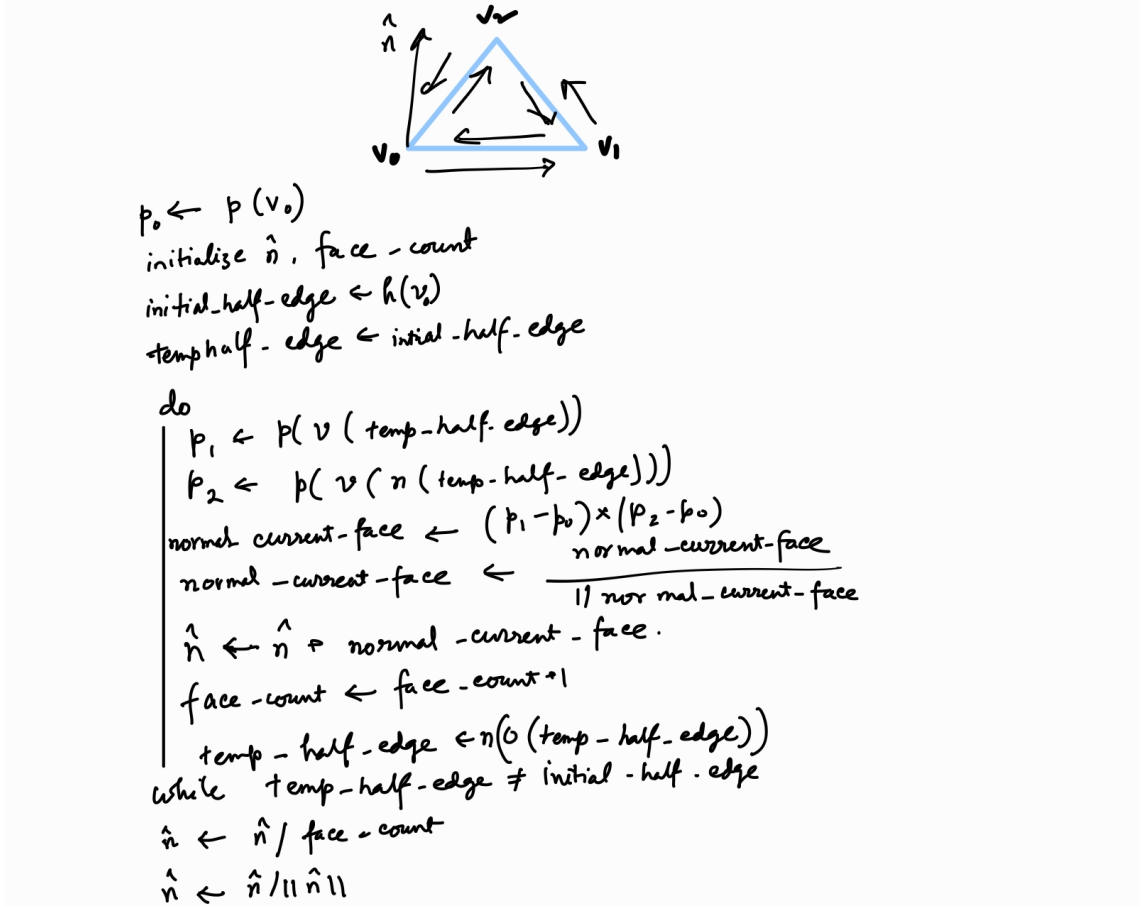
(a)



As seen from the figure above, a sequence of operations that navigates from vertex v_0 to v_7 is: $v(n(o(h(v(o(n(n(o(h(v_0))))))))))$.

(b) We want to implement a new operation $p(h) : H \to H$ that returns the previous halfedge of h (i. e. the halfedge h' such that $n(h') = h$). By the definition of $n(h) : H \to H$ it returns the next halfedge following h inside the same face (in counterclockwise direction). So for a triangle mesh without boundaries, we can go to the next half-edge in the same face in counterclockwise direction using $n(n(h))$ and then the next halfedge in the same face in counterclockwise direction will again be h (3 edges per triangular face), so we have $n(n(n(h)) = h$ and hence $h' = n(n(h))$ giving us $p(h) : H \to H = n(n(h))$.

(c) Given an additional operation $p(v) : V \to \mathbb{R}^3$ returning the 3D position of a vertex v, we can use the following pseudo-code that computes a vertex normal for a given vertex v by averaging the normals of the incident faces:



N.B. Initial values of face_count and normal are zero (or zero vector).