LAB 5: Delaunay triangulation

Joseba Sierra
18/12/2020

Updating the DCEL

Every time we add a new point during our incremental triangulation, we check for each adjacent triangle its opposite triangle, if the circle defined by it includes the inserted point, then we need to rotate the opposite edge (as seen in figure 1), to transform the current triangulation to a delaunay one. We repeat the process for each adjacent triangle.

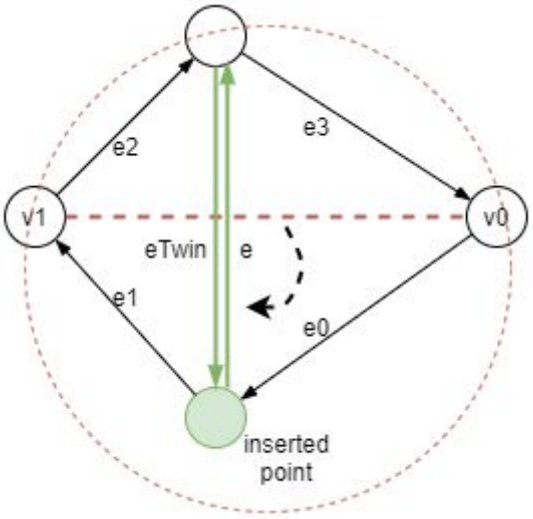

Figure 1: Case where edge rotation needed

As always, we need to update the DCEL to represent the new state. In this case, we need to update the rotated edge(‘e’ and ‘eTwin’), ‘e0’, ‘e1’, ‘e2’, ‘e3’, ‘v0’, ‘v1’, and both faces.

In this implementation we don’t need to worry about collinear and concyclic points. The triangulation being delaunay guarantees that collinear points will be triangulated correctly, due to the edge where the point falls being rotated. About the concyclic points, the way we add points guarantees that all faces are triangles, with no ambiguities.

Pruning boundaries

For each point of a boundary, we remove all incident edges (and their adjacent faces) of the edges that have their origin vertex outside the polygon defined by the boundary. To detect these vertices, we need to distinguish between 2 cases, as we can see in figure 2.
In the case of a left turn (boundary is given counterclockwise) those points that need to be removed are inside the UNION of the half planes defined by the 2 segments(1-2, 2-3). In the

right turn case, the zone that defines these points is the INTERSECTION of both half planes segments. For instance, if we need to know if a point ‘p’ is inside the right half plane defined by segment 1-2, is equivalent to say if det(1,2,p) < 0. This can also be used to know if 1,2,3 turn left or right (det(1,2,3) > 0 left and < 0 right). If det(1,2,3) == 0 the union and intersection of the half planes is equivalent, so we can just include it in one of the other cases.

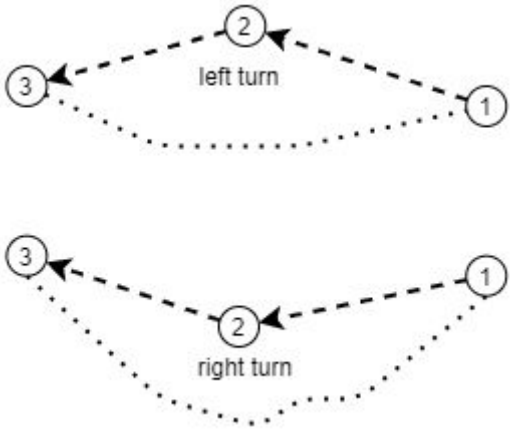

Figure 2: Both boundary cases