Geometric Algorithms
Assignment 4

Laura Baakman (s1869140)

October 15, 2014

A

A.1 Creating the DCEL

To create a DCEL one needs a couple of base classes, namely `Vertex`, `HalfEdge` and `Face`.

Vertex

The definition of the class `Vertex` and its methods that are pertinent now are presented in Listing 1. In line with the provided definition a `Vertex` has a set of coordinates representing its location, `coordinates`, and an edge, `incident_edge`, that has the `Vertex` as its origin.

I have overridden the equality definition since using the default definition could lead to infinite recursion. As the default compares all attributes of the class, which would mean comparing two `Vertex`'s and two `HalfEdges`. Comparing two `HalfEdges` means comparing, among others, its `origins` which would lead to comparing two `Vertexs` which would lead to comparing two `HalfEdges` and so on and so forth.

Since each `Vertex` is uniquely defined by its `coordinates` we only compare those when checking the equality of two `Vertexs`.

Listing 1: The definition of the class `Vertex()`.

```
    def __init__(self, coordinates, incident_edge=None):
        """Construct a Vertex object."""
        super(Vertex, self).__init__()
        self.coordinates = coordinates
        self.incident_edge = incident_edge

    def __eq__(self, other):
        """Chekc if two objects are equal by comparing only their coordinates."""
        if type(other) is type(self):
            return self.coordinates == other.coordinates
        return False

    def __neq__(self, other):
        """Check if two objects are not equal."""
        return not self.__eq__(other)
```

HalfEdge

A `HalfEdge` is a directed edge which is represented by its `origin`, a `Vertex` and a `twin`, which is the `HalfEdge` with this `HalfEdge`'s origin as its destination and this `HalfEdge`'s destination as its origin. The implementation of the class `HalfEdge` and the methods relevant to this discussion are presented in Listing 2.

Furthermore each `HalfEdge` stores an incident face and a next and previous edge. The `incident_face` is the face that is to the left-handed side when walking along this edge. The attributes `nxt` represent the edge one should take on arriving on the `HalfEdge`'s destination

when traversing the boundaries of its `incident_face`, `prev` is the `HalfEdge` one came from on the same walk.

The method `get_destination` returns the destination of the `HalfEdge` this is the same as the `origin` of its `twin`.

The `__eq__` method of `HalfEdge` has been overridden to avoid infinite recursion when comparing `HalfEdge`'s and to make it possible to compare an `HalfEdge` without the `nxt`, `incident_face` or `prev` property. This will not lead to any problems since an `HalfEdge` is uniquely defined by its origin and destination.

Listing 2: The definition of the class `Vertex()`.

```
    def __init__(self, origin, twin=None, incident_face=None, nxt=None, prev=None):
        """Construct a HalfEdge object."""
        super(HalfEdge, self).__init__()
        self.origin = origin
        self.twin = twin
        self.incident_face = incident_face
        self.nxt = nxt
        self.prev = prev

    def get_destination(self):
        """Return the destination of the halfedge as a vertex."""
        return self.twin.origin

    def __eq__(self, other):
        """Check if two half edges are equal."""
        if type(other) is type(self):
                return (
                    self.origin == other.origin and
                    self.twin.origin == other.twin.origin
                )
        return False

    def __neq__(self, other):
        """Check if two objects are not equal."""
        return not self.__eq__(other)
```

Introduceer de vertex, edge en face

Geef algemene algoritme voeg driehoek na driehoek toe

Driehoek toevoegen

Containing plane toevoegen

Geometrische operaties nodig gehad?

A.2 Walking Along the Outer Boundary

Bespreek number of vertices en get Edges in face

Highlight segments -> plaatje

Plot punten en de outer boundary van de DCEL en plot alle lijnen tussen alle punten om te checken of de outer boundary de convex hull is