Geometric Algorithms
Assignment 3

Laura Baakman (s1869140)

September 27, 2014

A

Point in Triangle

We try to determine if the point $\mathbf{Q}$ lies in the triangle defined by the points $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$. To this end we define the vectors $\mathbf{v}_1 = \mathbf{P_2} - \mathbf{P_1}$ and $\mathbf{v_2} = \mathbf{P_3} - \mathbf{P_1}$, see Figure 1. Each point $\mathbf{P}$ inside the grey area in this figure can be described as:

$$\mathbf{P} = \mathbf{P_1} + a \cdot \mathbf{v_1} + b \cdot \mathbf{v_2}. \tag{1}$$

For all points to the right of $\mathbf{P_1}$ $a > 0$ and $b > 0$. The points that define the triangle can all be expressed according to (1):

$$\mathbf{P_1} = \mathbf{P_1} + 0 \cdot \mathbf{v_1} + 0 \cdot \mathbf{v_2} \tag{2}$$
$$\mathbf{P_2} = \mathbf{P_1} + 1 \cdot \mathbf{v_1} + 0 \cdot \mathbf{v_2} \tag{3}$$
$$\mathbf{P_3} = \mathbf{P_1} + 0 \cdot \mathbf{v_1} + 1 \cdot \mathbf{v_2}. \tag{4}$$

If $a + b == 1$ the point lies on the edge between $\mathbf{P_2}$ and $\mathbf{P_3}$. Based on Equation 2 through 4 we find that a point $\mathbf{Q}$ lies on the triangle if it can be expressed according to (1) with $a, b \in (0, 1)$ and with $a + b < 1$.

Solving the resulting equation with Mathematica, see Listing 1, gives us expressions for a and b, namely:

$$a = -\frac{-\mathbf{P_{1,0}P_{3,1}} + \mathbf{P_{1,0}Q1} + \mathbf{P_{1,1}P_{3,0}} - \mathbf{P_{1,1}Q_0} - \mathbf{P_{3,0}} \cdot \mathbf{Q_1} + \mathbf{P_{3,1}} \cdot \mathbf{Q_0}}{-\mathbf{P_{1,0}P_{2,1}} + \mathbf{P_{1,0}} \cdot \mathbf{P_{3,1}} + \mathbf{P_{1,1}} \cdot \mathbf{P_{20}} - \mathbf{P_{1,1}P_{3,0}} - \mathbf{P_{2,0}} \cdot \mathbf{P_{3,1}} + \mathbf{P_{2,1}} \cdot \mathbf{P_{3,0}}} \tag{5}$$
$$b = -\frac{-\mathbf{P_{1,0}} \cdot \mathbf{P_{2,1}} + \mathbf{P_{1,0}} \cdot \mathbf{Q_1} + \mathbf{P_{1,1}} \cdot \mathbf{P_{2,0}} - \mathbf{P_{1,1}Q_0} - \mathbf{P_{2,0}} \cdot \mathbf{Q_1} + \mathbf{P_{2,1}} \cdot \mathbf{Q_0}}{\mathbf{P_{1,0}P_{2,1}} - \mathbf{P_{1,0}} \cdot \mathbf{P_{3,1}} - \mathbf{P_{1,1}} \cdot \mathbf{P_{2,0}} + \mathbf{P_{1,1}P_{3,0}} + \mathbf{P_{2,0}} \cdot \mathbf{P_{3,1}} - \mathbf{P_{2,1}} \cdot \mathbf{P_{3,0}}} \tag{6}$$

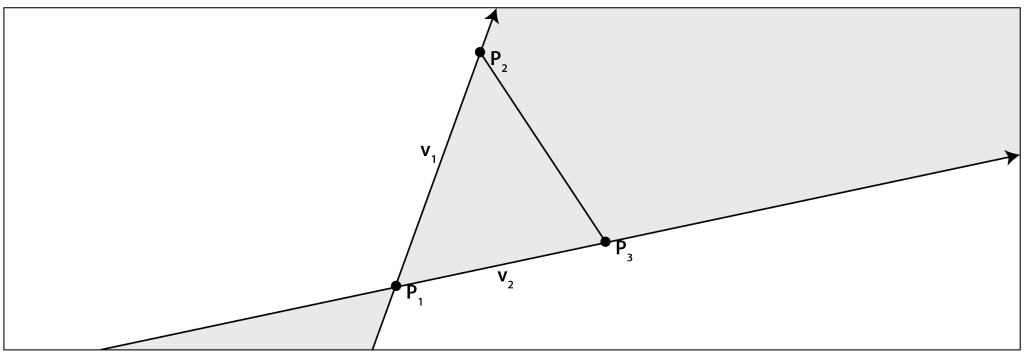

Figure 1: A triangle defined by the points $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$, with the vectors $\mathbf{v_1}$ and $\mathbf{v_2}$. The grey area covers all points that can be described according to (1).

Listing 1: Mathematica code used to compute the to compute a and b.

```
p1 = {p10, p11};
p2 = {p20, p21};
p3 = {p30, p31};
v1 = p2 - p1;
v2 = p3 - p1;

p4 = p1 + a * v1 + b * v2;
p41 = Part[p4, 1] == Q0;
p42 = Part[p4, 2] == Q1;
solution = Solve[{p41, p42}, {a, b}]
```

Listing 2: The method `point_in_triangle()` in the module `triangle`.

```
def point_in_triangle(triangle, point):
    """
    Return true if the point lies in the triangle.

    Input:
        triangle: List of three points, where each point is a list
            with the x and y coordinate of an vertex of the triangle.
        point: List with the x and y coordinate of the point.
    """
    [p1, p2, p3] = triangle
    v1_cross_v2 = (
        -p1[1] * p2[0] + p1[0] * p2[1] + p1[1] * p3[0]
        - p2[1] * p3[0] - p1[0] * p3[1] + p2[0] * p3[1]
    )
    if(v1_cross_v2):
        a_numerator = (
            p1[1] * p3[0] - p1[0] * p3[1] - p1[1] * point[0] +
            p3[1] * point[0] + p1[0] * point[1] - p3[0] * point[1]
        )
        a = a_numerator / v1_cross_v2
        if(a > 0 and a < 1):
            b_numerator = (
                p1[1] * p2[0] - p1[0] * p2[1] - p1[1] * point[0] +
                p2[1] * point[0] + p1[0] * point[1] - p2[0] * point[1]
            )
            b = - b_numerator / v1_cross_v2
            return (b > 0 and b < 1) and (a + b < 1)
    return False
```

where $\mathbf{P_{r,s}}$ represents the s'th element of the point r and $\mathbf{Q_t}$ represent the t'th element of the point $\mathbf{Q}$.

The denominator of (5) and (6) are the same, this is the magnitude of $\mathbf{v_1} \times \mathbf{v_2}$ where the vectors are made three-dimensional by adding a z-coordinate of zero, If that cross product is zero $\mathbf{v_1}$ and $\mathbf{v_2}$ are parallel and (1) can thus be only used to represent points on a line parallel to $\mathbf{v_1}$ and through $\mathbf{P_1}$.

The method presented above is implemented in the method `point_in_triangle()` in the module `triangle`, see Listing 2.

A.1 Finding the Triangle Containing the Point

We have simply checked for all triangles that result from the triangulation if the point lies inside that triangle, see Listing 3.

B

Listing 3: The method `find_containing_triangle()`.

```
    cens:   Array with a list of list where each sublist contains the coordinates
            of center of one of the triangles of the triangulation.
    edges:  Array with a list of list where each sublist contains the indices
            of the points between which one of the edges of the triangulation runs.
    triPts: Array with triangles, each triangle is represented as a list of three
            indices into xa and ya.
    neighs:  Array of integers giving the indices into cens
            triPts, and neighs of the neighbors of each trianglegit
"""

from random import *
import matplotlib.delaunay as triang
import numpy
import pdb


try:
    from OpenGL.GLUT import *
    from OpenGL.GL import *
    from OpenGL.GLU import *
except:
    print '''ERROR: PyOpenGL not installed properly.'''
    print '''Go get it: http://atrpms.net/'''
    exit(2)

width = 850
height = 850
points = []
```