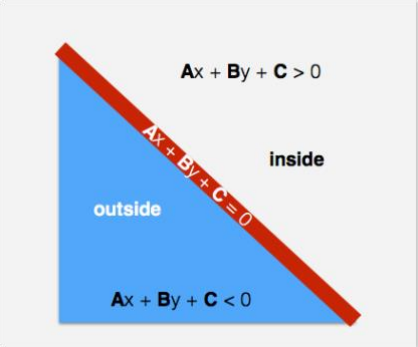
Laboratory work 10

1 Objectives

This laboratory presents the topic of triangle rasterization using barycentric coordinates.

2 Triangle definitions

We define each triangle by edges and we compute the edge equations such that the negative halfplane to be on the triangle's exterior. We start from the general implicit form of a line (in 2D) $Ax + By + C = 0$. The implicit form of a line passing through points A(x_a, y_a) and B(x_b, y_b) is: $(y_a - y_b)x + (x_b - x_a)y + x_a y_b - x_b y_a = 0$.

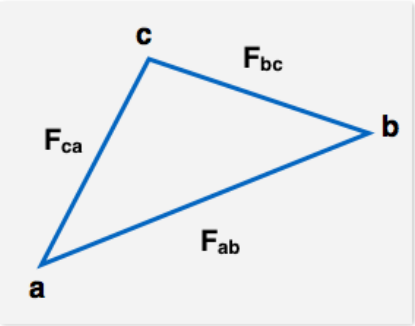


For a triangle define by vertices a(x_a, y_a), b(x_b, y_b) and c(x_c, y_c) we have the following edge equations:

$$F_{ab} = (y_a - y_b)x + (x_b - x_a)y + x_a y_b - x_b y_a = 0$$

$$F_{bc} = (y_b - y_c)x + (x_c - x_b)y + x_b y_c - x_c y_b = 0$$

$$F_{ca} = (y_c - y_a)x + (x_a - x_c)y + x_c y_a - x_a y_c = 0$$



2.1 Barycentric coordinates

Any point p is a linear combination of points P, Q, and R:

$$\begin{aligned} p &= P + \beta(Q - P) + \gamma(R - P) \\ &= (1 - \beta - \gamma)P + \beta Q + \gamma R \\ &= \alpha P + \beta Q + \gamma R \end{aligned}$$

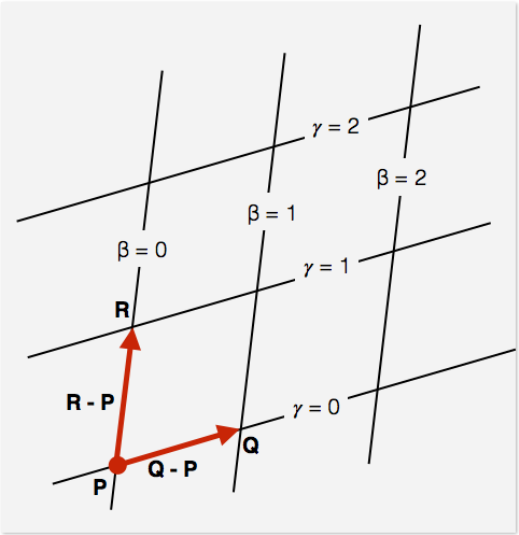
For triangles we need that:

$$\beta + \gamma \leq 1$$

$$\beta \geq 0$$

$$\gamma \geq 0$$

$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ are called the **barycentric coordinates**.



Barycentric coordinates describe a point p as an affine combination of the triangle vertices:

$p = \alpha P + \beta Q + \gamma R$, where $\alpha + \beta + \gamma = 1$

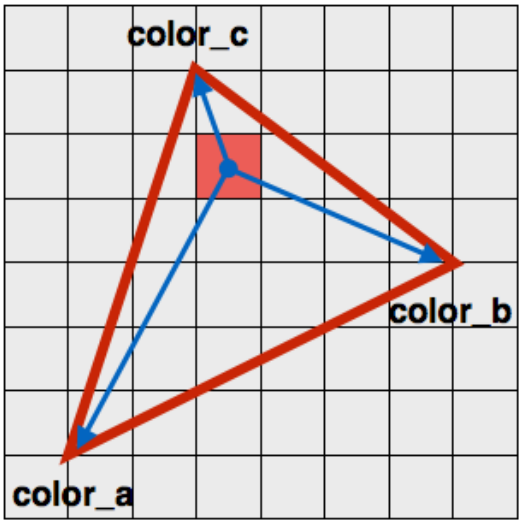
For any point p inside the triangle specified by vertices a, b, and c:

$0 < \alpha < 1$

$0 < \beta < 1$

$0 < \gamma < 1$

A point is on an edge if one barycentric coordinate is 0, and a point is on a vertex if two barycentric coordinates are 0.



For a triangle the barycentric coordinates for point p are:

$$\alpha = \frac{F_{bc}(x,y)}{F_{bc}(x_a,y_a)}$$

$$\beta = \frac{F_{ac}(x,y)}{F_{ac}(x_b,y_b)}$$

$$\gamma = 1 - \alpha - \beta$$

2.2 Triangle rasterization pseudocode

```
void triangleRasterization(vertices v[3]){
    bbox b = findBoundingBox(v);
    foreach pixel(x, y) in b
    {
      compute alpha, beta, gamma;
      if(0 < alpha < 1 and
        0 < beta < 1 and
        0 < gamma < 1)
      {
        color = color_a * alpha + color_b * beta + color_c * gamma;
        drawPixel(x, y) with color;
      }
    }
}
```

3 Assignment

- Extend the implementation from the previous laboratory to add the functionality of displaying the 3D object filled with color.