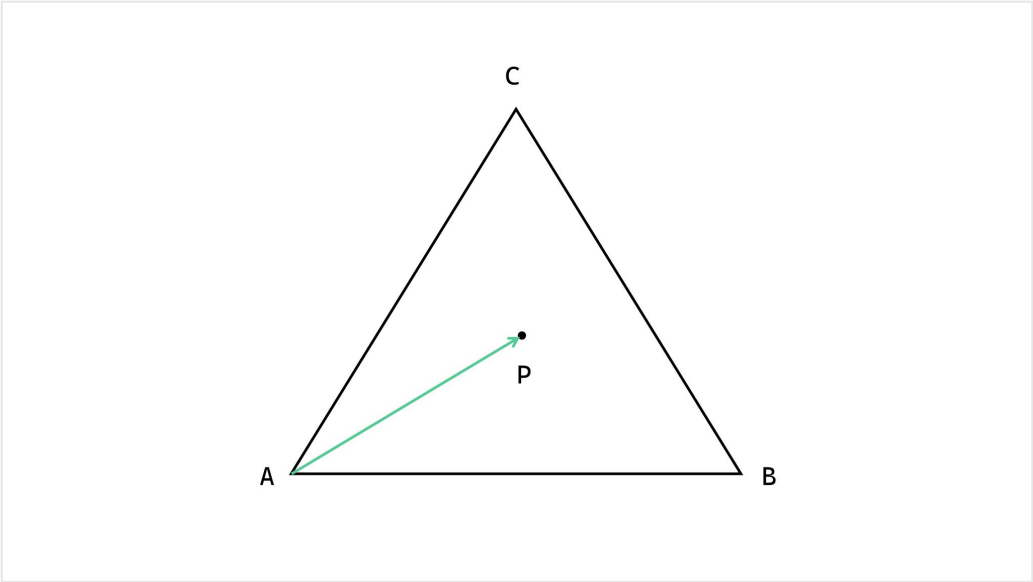
Note: Block-based Half-Space Triangle Rasterization

Fong Zi-Sing, 2022.3

---

1. Edge function and barycenteric interpolation[1]



- If the vertices A, B, C are given in counterclockwise order, then

  " P is inside the triangle "

  is equivalent to

$$\left|\overrightarrow{AB} \times \overrightarrow{AP}\right| > 0, \quad \left|\overrightarrow{BC} \times \overrightarrow{BP}\right| > 0, \quad \left|\overrightarrow{CA} \times \overrightarrow{CP}\right| > 0$$

- And it can be described as:

$$I_{01} = (A_y - B_y),\ J_{01} = (B_x - A_x),\ K_{01} = (A_x B_y - A_y B_x)$$
$$I_{02} = (B_y - C_y),\ J_{02} = (C_x - B_x),\ K_{02} = (B_x C_y - B_y C_x)$$
$$I_{03} = (C_y - A_y),\ J_{03} = (A_x - C_x),\ K_{03} = (C_x A_y - C_y A_x)$$
$$\left|\overrightarrow{AB} \times \overrightarrow{AP}\right| = F_{01}(P_x, P_y) = I_{01} \cdot P_x + J_{01} \cdot P_y + K_{01} \quad (1)$$
$$\left|\overrightarrow{BC} \times \overrightarrow{BP}\right| = F_{02}(P_x, P_y) = I_{02} \cdot P_x + J_{02} \cdot P_y + K_{02} \quad (2)$$
$$\left|\overrightarrow{CA} \times \overrightarrow{CP}\right| = F_{03}(P_x, P_y) = I_{03} \cdot P_x + J_{03} \cdot P_y + K_{03} \quad (3)$$

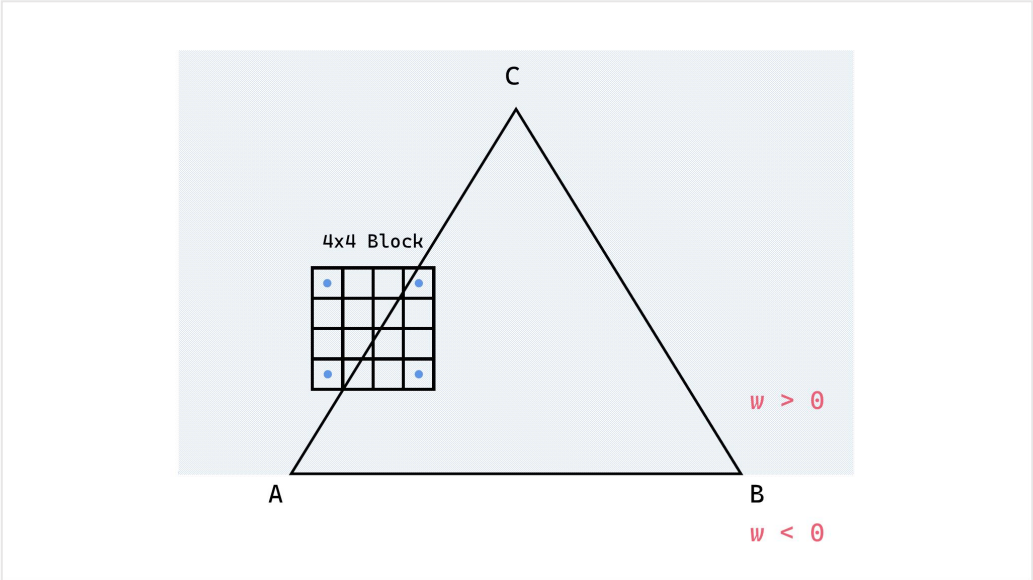
- Then, the interpolation coefficients are the followings:

$$u = \frac{F_{02}}{F_{01}+F_{02}+F_{03}} = \frac{S_{\Delta BCP}}{S_{\Delta ABC}} \quad (4)$$

$$v = \frac{F_{03}}{F_{01}+F_{02}+F_{03}} = \frac{S_{\Delta CAP}}{S_{\Delta ABC}} \quad (5)$$

$$w = \frac{F_{01}}{F_{01}+F_{02}+F_{03}} = \frac{S_{\Delta ABP}}{S_{\Delta ABC}} \quad (6)$$

2. Block-based



- For the four corner of a block, there are three cases:
  - every corner of the block is outside the triangle.
  - every corner of the block is inside the triangle.
  - the block overlaps the triangle.

- If only the $|\overrightarrow{AB} \times \overrightarrow{AP}|$ is considered, for the every corner of a block:
  - outside ⇔ all $w < 0$
  - inside ⇔ all $w > 0$
  - overlap ⇔ some corner $w < 0$, other corner $w > 0$

- The same as $|\overrightarrow{BC} \times \overrightarrow{BP}|$, $|\overrightarrow{CA} \times \overrightarrow{CP}|$.

- On this basis, the pseudo code is the following:

```
q = GetBlockSize()
[ minX, maxX, minY, maxY ] = CalcuteBoundingBox(triangle.vertices[3],screenSize)
[ I01, I02, I03 ] = CalcuteDeltaX(triangle.vertices[3])
[ J01, J02, J03 ] = CalcuteDeltaY(triangle.vertices[3])
[ K01, K02, K03 ] = CalcuteConst(triangle.vertices[3])
[ I00, J00, K00 ] = CalcuteConst(triangle.vertices[3],
                                 I01, I02, I03,
                                 J01, J02, J03,
                                 K01, K02, K03)


/** __mm128 is SIMD struct, details see `intel.com` */
I = Make __mm128(I00, I01, I02, I03)
J = Make __mm128(J00, J01, J02, J03)
K = Make __mm128(K00, K01, K02, K03)

F = I * minX + J * minY + K;

Area2 = F[1] + F[2] + F[3]
A = Make __mm128(Area2, Area2, Area2, Area2)

/** ( 1 / depth, gamma, alpha, beta ) at ( minX, minY ) */
F = F / A

/** dx, dy */
I = I / A, J = I / A

/** the four corner of a block */
[ F1, F2, F3, I1, I2, I3, J1, J2, J3 ] = CalculateFourCorner(F, I, J, q);

I *= q, J *= q, Cy = F, Cy1 = F1, Cy2 = F2, Cy3 = F3
loop y = minY to maxY with step q
{
    Cx = Cy, Cx1 = Cy1, Cy2 = Cx2, Cx3 = Cy3
    loop x = minX to maxX with step q
    {
        /** check w of the four corner */
        checkw = (Cx1[0] > 0) << 3 |
                 (Cx1[1] > 0) << 2 |
                 (Cx1[2] > 0) << 1 |
                 (Cx1[3] > 0)

        /** the same as u, v */
        checku = (Cx2[0]>0)<<3 | (Cx2[1]>0)<<2 | (Cx2[2]>0)<<1 | (Cx2[3]>0)
        checkv = (Cx3[0]>0)<<3 | (Cx3[1]>0)<<2 | (Cx3[2]>0)<<1 | (Cx3[3]>0)


        /** Fully outside */
        if (checkw == 0x0 || checku == 0x0 || checkv == 0x0)
        {
          Cx += I, Cx1 += I1, Cx2 += I2, Cx3 += I3
          continue;
        }

        /** Fully inside */
        if (checkw == 0xF || checku == 0xF || checkv == 0xF)
        {
          RenderBlock(x, y, Cx)
        }
        /** Overlaps */
        else
        {
          loop by = 0 to q with step 1
          {
            loop by = 0 to q with step 1
            {
              RenderPixel(x+bx, y+by, Cx)
            }
          }
        }
        Cx += I, Cx1 += I1, Cx2 += I2, Cx3 += I3
    }
    Cy += J, Cy1 += J1, Cy2 += J2, Cy3 += J3
}
```

References

[1] Mileff P , K Nehéz , Dudra J . *Accelerated Half-Space Triangle Rasterization*[J]. Acta Polytechnica Hungarica, 2015, 12(7):2015-2217.