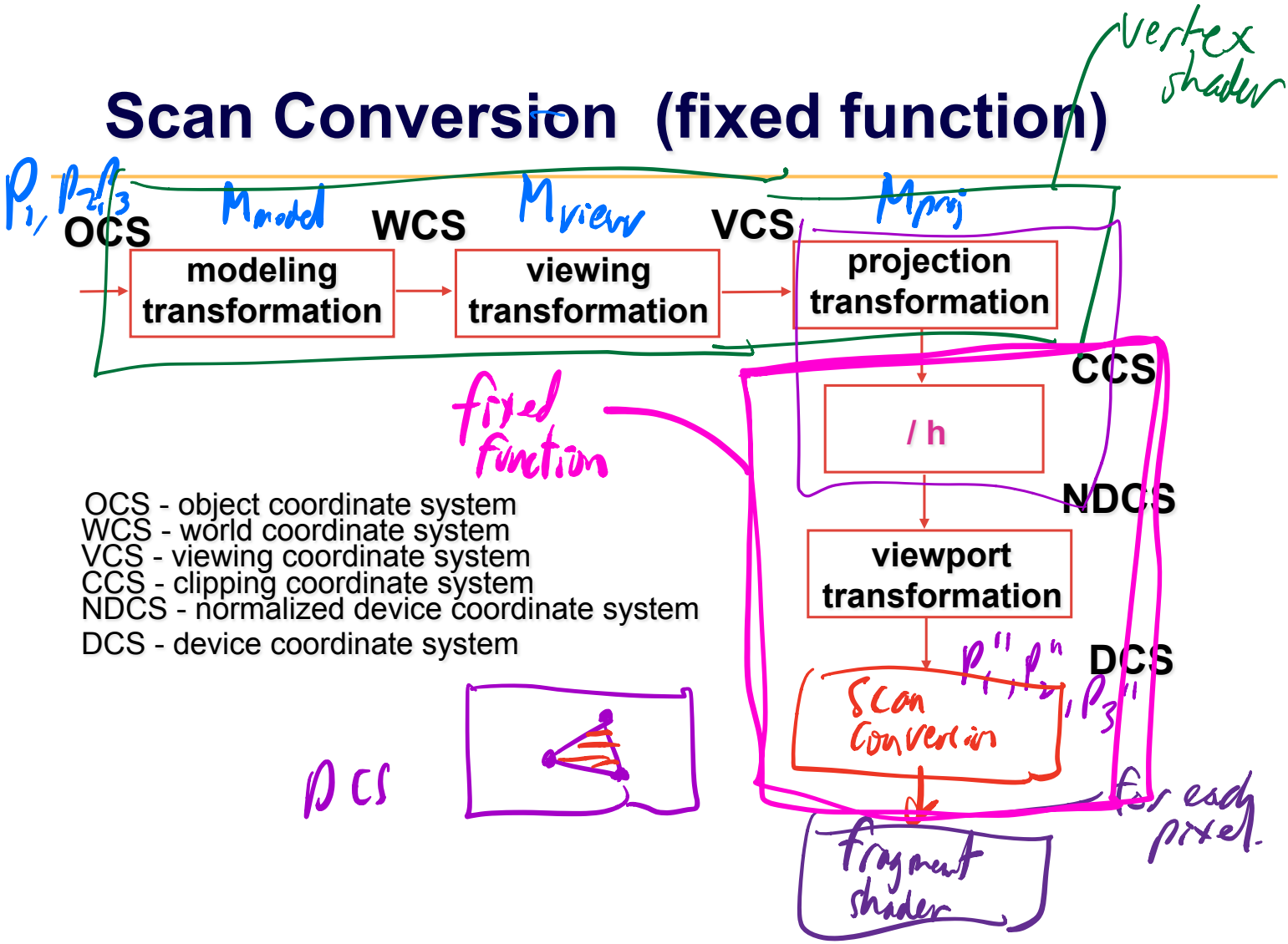
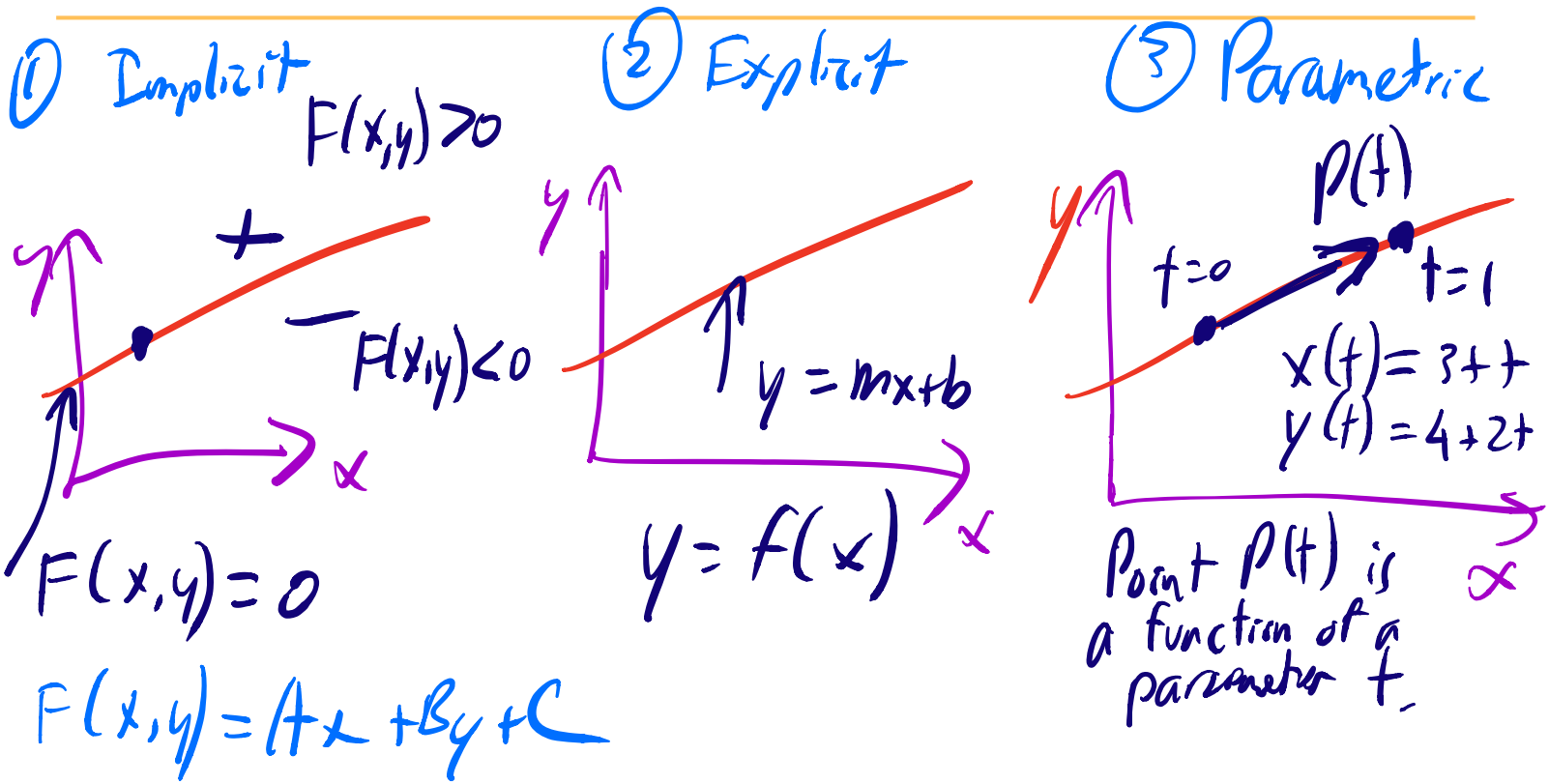
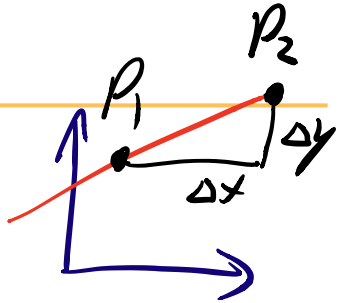
Scan Conversion (fixed function)



Implicit, Explicit, and Parametric equations for defining geometry



Lines and Curves



Explicit

line

$y = mx + b$

$y = y_1 + m\Delta x$

$y = y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1)$

circle

$y = \pm\sqrt{r^2 - x^2}$ $\qquad x^2 + y^2 = r^2$

plane

$z = Ax + By + C$

sphere

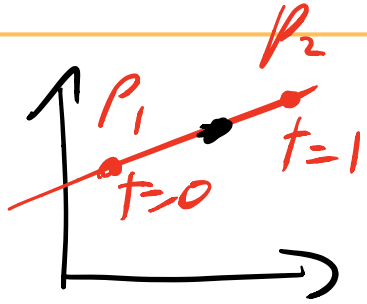
$z = \pm\sqrt{r^2 - x^2 - y^2}$

Lines and Curves

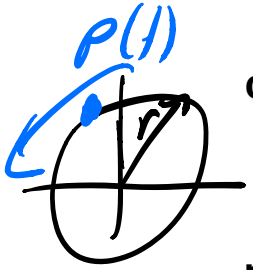
Parametric

line

$$p(t) = p_1 + t(p_2 - p_1)$$
$$= p_1(1-t) + t p_2$$



basis functions

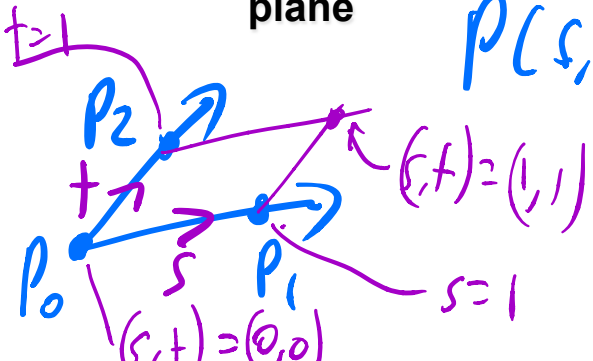


circle

$$x(t) = r\cos(t)$$
$$y(t) = r\sin(t)$$
$$t \in [0, 2\pi]$$

plane

$$p(s,t) = p_0 + s(p_1 - p_0) + t(p_2 - p_0)$$



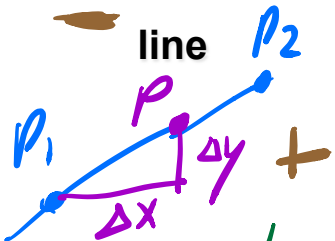
$$\begin{bmatrix} x(s,t) \\ y(s,t) \\ z(s,t) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + s\begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \\ z_1 - z_0 \end{bmatrix} + t\begin{bmatrix} x_2 - x_0 \\ y_2 - y_0 \\ z_2 - z_0 \end{bmatrix}$$

Lines and Curves

Implicit

2D $F(x,y)=0$

3D $F(x,y,z)=0$



$y = mx+b$

$0 = -y + mx + b$

circle

$r^2 = x^2 + y^2$

$0 = x^2 + y^2 - r^2$

$$y = y_1 + \frac{(y_2-y_1)}{(x_2-x_1)}(x-x_1)$$

$$0 = (y_1-y) + \frac{(y_2-y_1)}{(x_2-x_1)}(x-x_1)$$

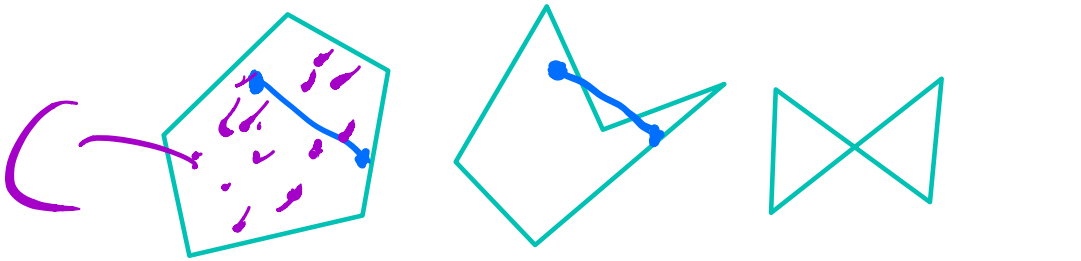
$$0 = (y_1-y)(x_2-x_1) + (y_2-y_1)(x-x_1)$$

$$0 = x(y_2-y_1) + y(x_1-x_2) + y_1x_2 - x_1y_1 - x_1y_2 + x_1y_1$$

$$0 = Ax + By + C$$

Polygons

Interactive graphics uses polygons



simple convex | **simple concave** | **non-simple (self-intersection)**

Simple: edges do not self intersect

Convex: interior angles $\theta_i \leq 180°$

set $C \subseteq \mathbb{R}^d$ is convex if

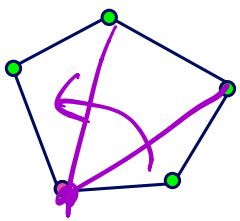
for any two points $P, Q \in C$ and any $\alpha \in [0,1]$

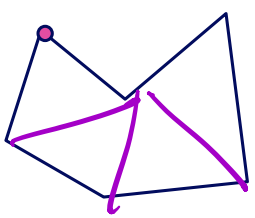
$\alpha P + (1-\alpha) Q \in C$

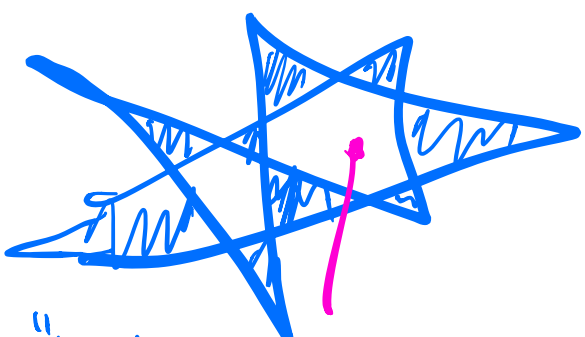
In practice we use triangles

- why? triangles are always: simple, convex, planar

- simple convex polygons
  - *trivial to break into triangles*
- concave or non-simple polygons
  - *more effort to break into triangles*

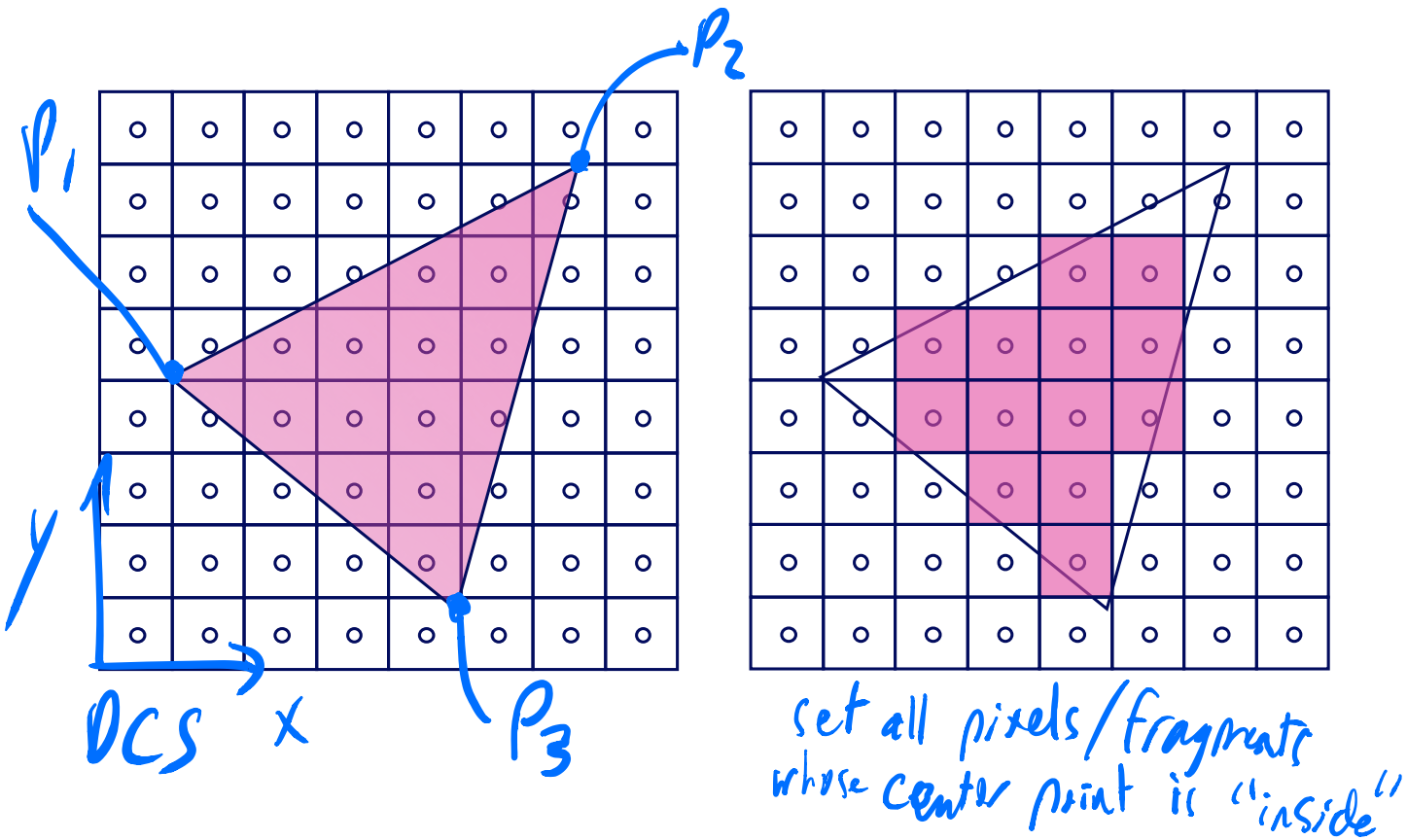






one option:
can use "winding number"

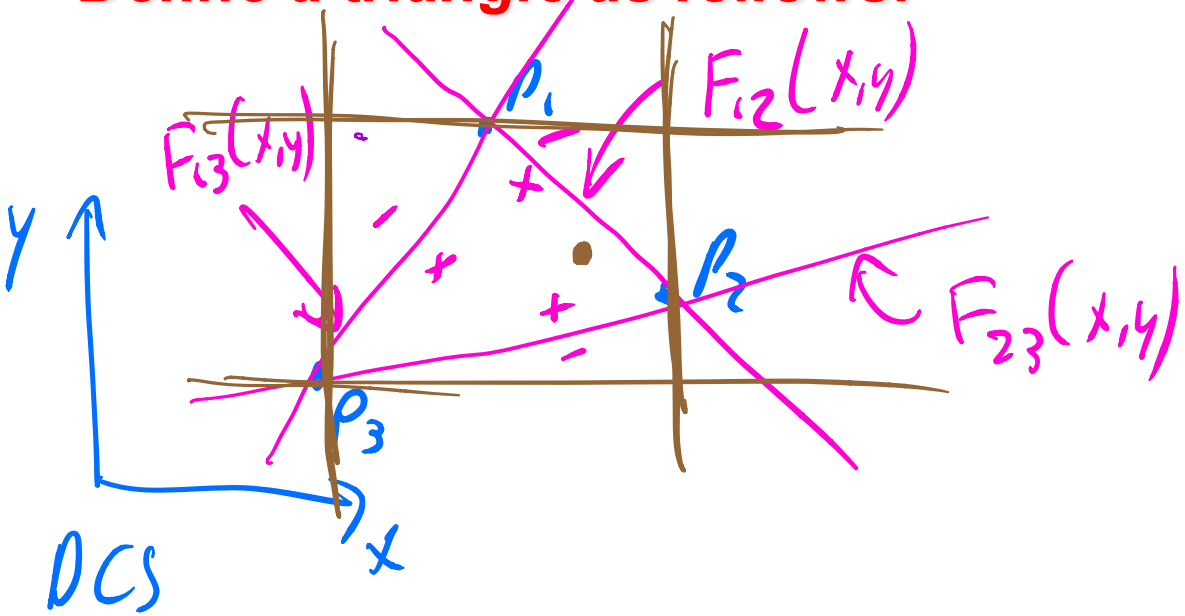
What is Scan Conversion? (a.k.a. Rasterization)



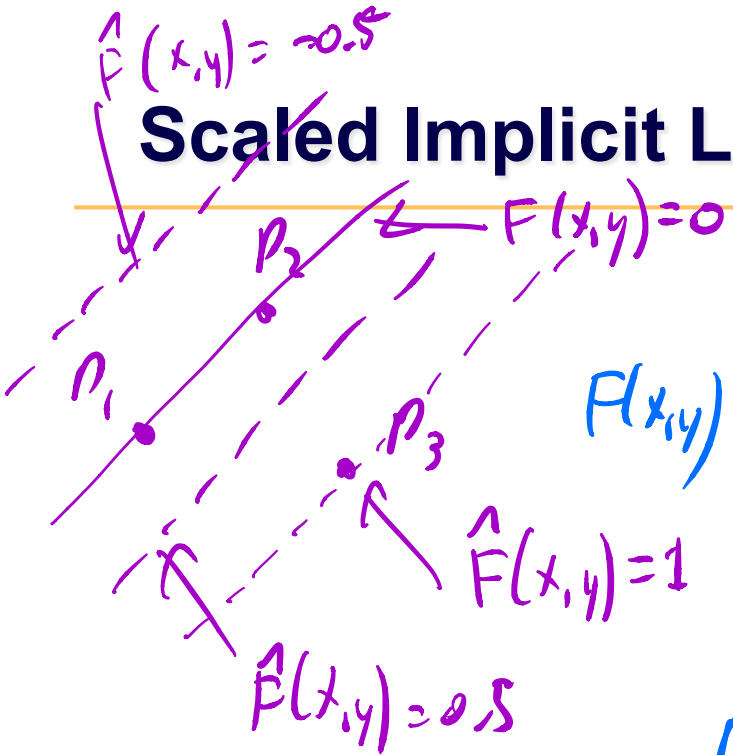
AABB: axis-aligned bounding box

Modern Rasterization

Define a triangle as follows:



Scaled Implicit Line Equation



$$0 = x(y_2 - y_1) + y(x_1 - x_2) + y_1x_2 - y_2x_1$$

$$F(x,y) = 0 = Ax + By + C$$

Now develop $\hat{F}(x,y)$ such that $\hat{F}(x_3,y_3) = +1$

Define $k = F(x_3,y_3)$

Then $\hat{F}(x,y) = \frac{F(x,y)}{k}$

i.e. $\hat{F}(x,y) = \left(\frac{A}{k}\right)x + \left(\frac{B}{k}\right)y + \frac{C}{k}$

Edge Equations: Code

```
findBoundingBox(&xmin, &xmax, &ymin, &ymax);
setupEdges (&a0,&b0,&c0,&a1,&b1,&c1,&a2,&b2,&c2);

for (int y = yMin; y <= yMax; y++) {
      for (int x = xMin; x <= xMax; x++) {
          float e0 = a0*x + b0*y + c0;
          float e1 = a1*x + b1*y + c1;
          float e2 = a2*x + b2*y + c2;
          if (e0 > 0 && e1 > 0 && e2 > 0)
                    Image[x][y] = TriangleColor;
      }
}
```

walk through pixels in bounding box

$F_{12}(x,y)$ $F_{23}(x,y)$ $F_{13}(x,y)$

"inside"

Edge Equations: Code

```
  // more efficient inner loop
for (int y = yMin; y <= yMax; y++) {
      float e0 = a0*xMin + b0*y + c0;
      float e1 = a1*xMin + b1*y + c1;
      float e2 = a2*xMin + b2*y + c2;
      for (int x = xMin; x <= xMax; x++) {
        if (e0 > 0 && e1 > 0 && e2 > 0)
          Image[x][y] = TriangleColor;
        e0 += a0;   e1+= a1;    e2 += a2;
      }
}
```

Interpolation During Scan Conversion

- interpolate values from vertices to interior pixels:
  - *z*
  - *r,g,b colour components*
  - *u,v texture coordinates*
  - N_x, N_y, N_z *surface normals*
- three equivalent ways of viewing this (for triangles)
  1. *bilinear interpolation*
  2. *plane equation*
  3. *barycentric coordinates*

Linear interpolation : LERP

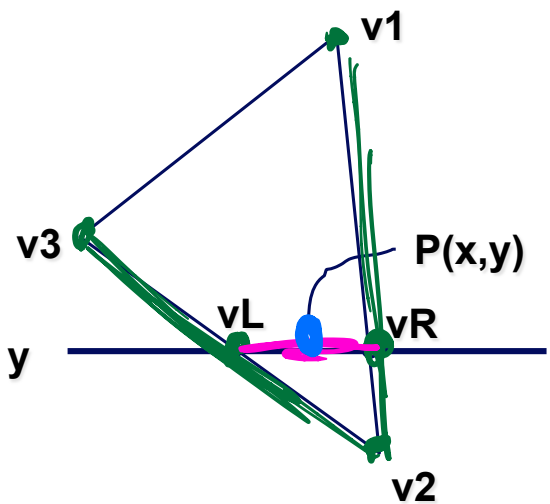
1. Bilinear Interpolation

- interpolate quantity along LH and RH edges, as a function of y
  - *then interpolate quantity as a function of x*

$V_L = f(y)$

$V_R = f(y)$

$V = f(x)$



fraction of the way towards P_3 from P_2

$$V_L = V_2 + \left(\frac{y - y_2}{y_3 - y_2}\right)(V_3 - V_2)$$

V_R = (similar)

$$V = V_L + \left(\frac{x - x_L}{x_R - x_L}\right)(V_R - V_L)$$

2. Plane Equation

- v = Ax + By + C

$$V_1 = Ax_1 + By_1 + C$$
$$V_2 = Ax_2 + By_2 + C$$
$$V_3 = Ax_3 + By_3 + C$$

} Solve for A, B, C

for any (x, y) pixel

$$V = Ax + By + C$$

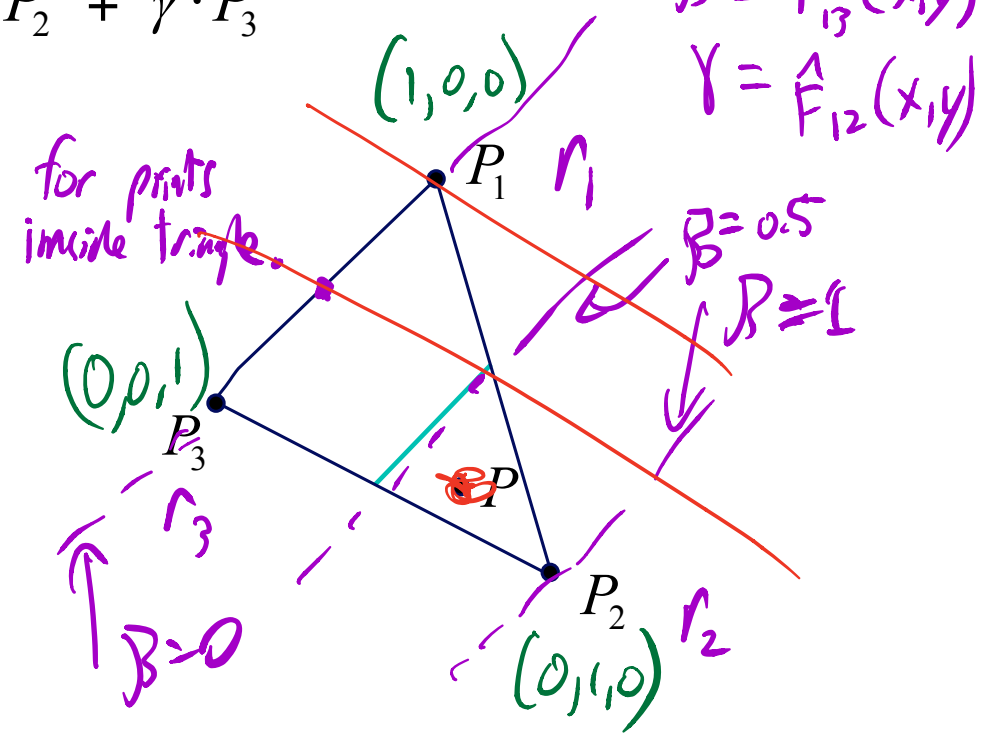
3. Barycentric Coordinates

- ***weighted combination of vertices***

$$P = \alpha \cdot P_1 + \beta \cdot P_2 + \gamma \cdot P_3$$

$$\alpha + \beta + \gamma = 1$$

$$0 \le \alpha, \beta, \gamma \le 1$$



Barycentric Coordinates

- once computed, use to interpolate any # of parameters from their vertex values

$$v = \alpha \cdot v_1 + \beta \cdot v_2 + \gamma \cdot v_3$$

$\alpha, \beta, \gamma \in [0,1]$

$\alpha + \beta + \gamma = 1$

- computing Barycentric coordinates

e.g.

$r = \alpha r_1 + \beta r_2 + \gamma r_3$

$g = \alpha g_1 + \beta g_2 + \gamma g_3$

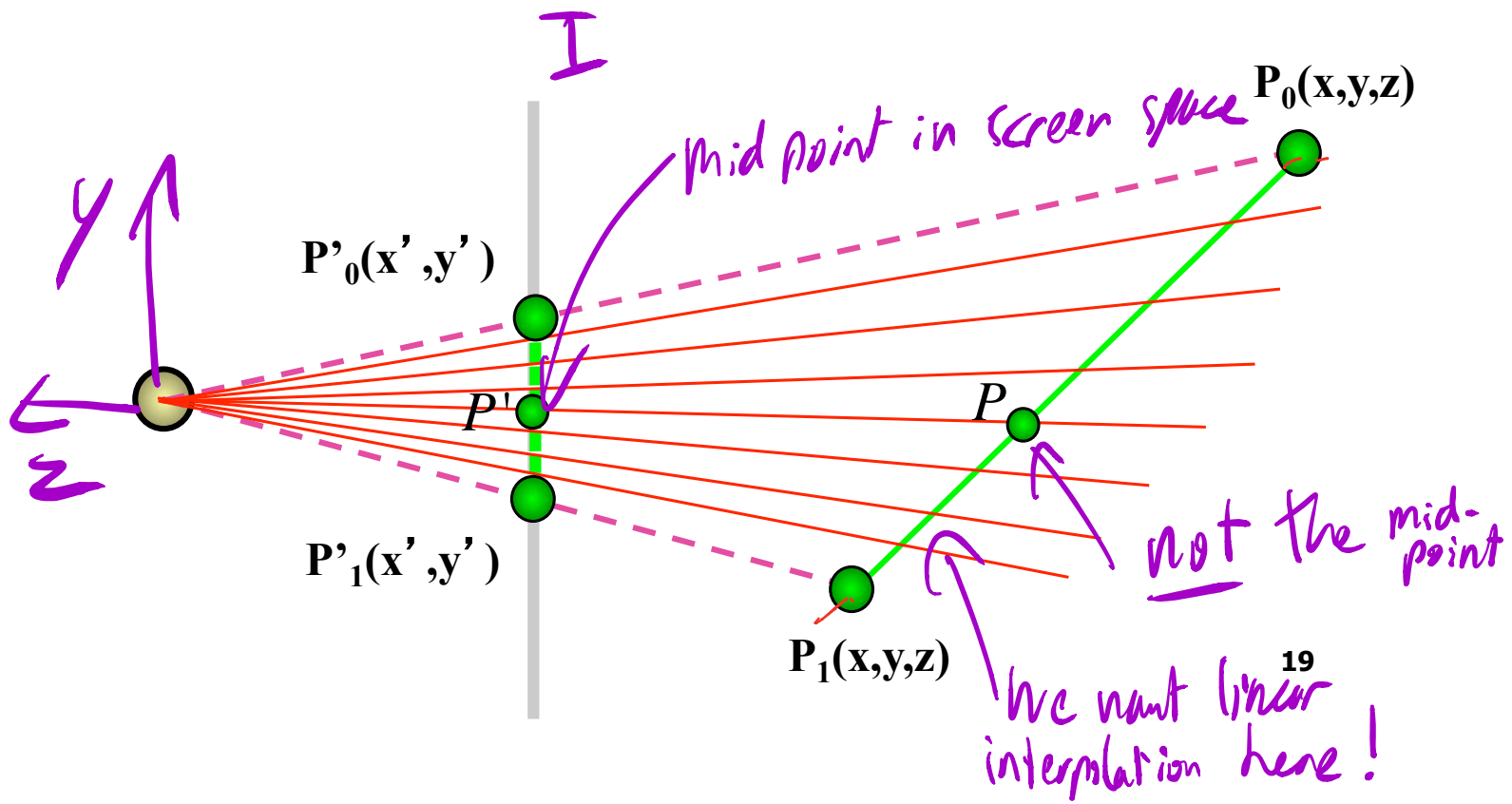
$b = \ldots$

$\alpha = \hat{F}_{23}(x,y)$, i.e., the scaled implicit line equation that passes through P_2P_3, i.e.,

$\hat{F}_{23}(x_2,y_2) = \hat{F}_{23}(x_3,y_3) = 0$

$\hat{F}_{23}(x_1,y_1) = 1$

Interpolation: Screen vs World Space



Perspective-correct interpolation

$$v = \frac{\alpha \cdot v_1 / h_1 + \beta \cdot v_2 / h_2 + \gamma \cdot v_3 / h_3}{\alpha / h_1 + \beta / h_2 + \gamma / h_3}$$

$$v = \frac{Barycentric(\frac{v_1}{h_1}, \frac{v_2}{h_2}, \frac{v_3}{h_3})}{Barycentric(\frac{1}{h_1}, \frac{1}{h_2}, \frac{1}{h_3})}$$

← h in CCS

$$= \frac{\alpha\left(\frac{v_1}{h_1}\right) + \beta\left(\frac{v_2}{h_2}\right) + \gamma\left(\frac{v_3}{h_3}\right)}{\alpha\left(\frac{1}{h_1}\right) + \beta\left(\frac{1}{h_2}\right) + \gamma\left(\frac{1}{h_3}\right)}$$