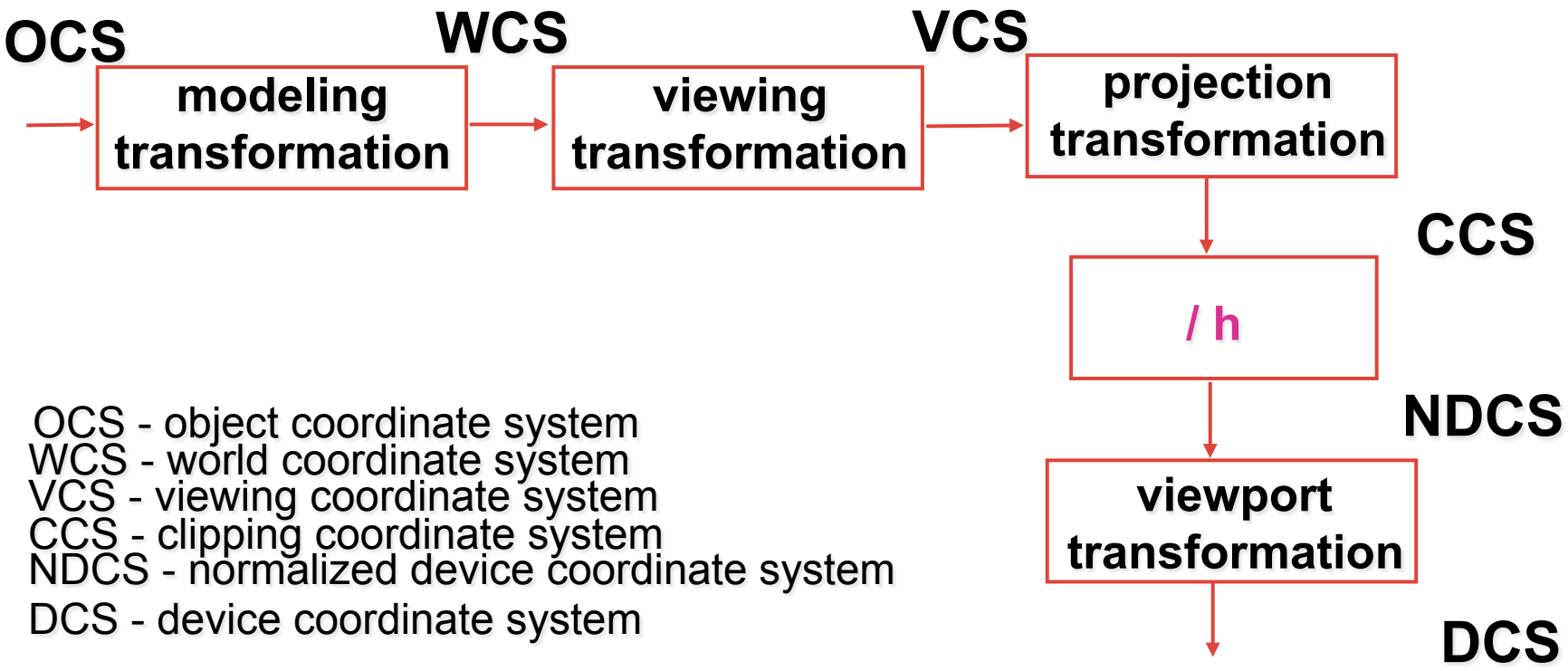
Scan Conversion (fixed function)



OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

Implicit, Explicit, and Parametric equations for defining geometry

Lines and Curves

Explicit

line

circle

plane

sphere

Lines and Curves

Parametric

line

circle

plane

Lines and Curves

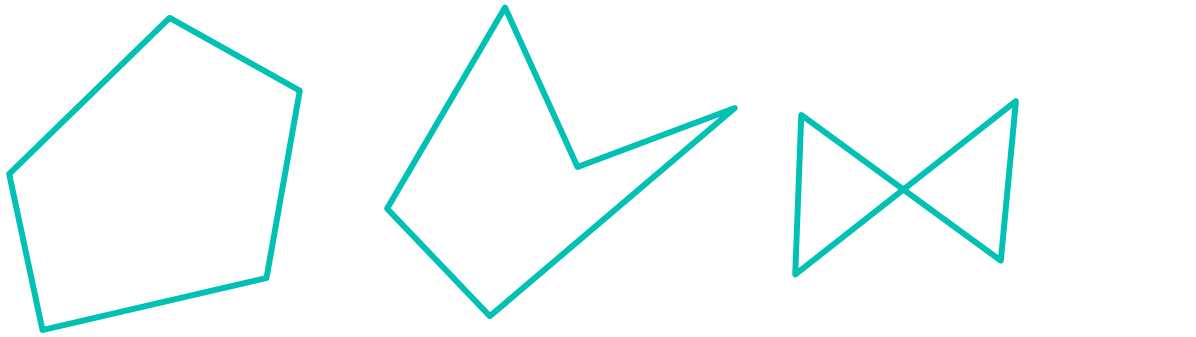
Implicit

line

circle

Polygons

Interactive graphics uses polygons



simple convex

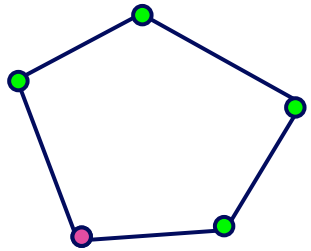
simple concave

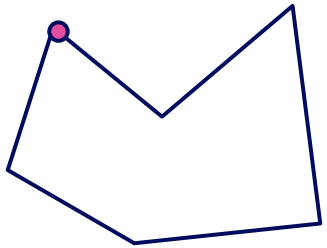
non-simple (self-intersection)

In practice we use triangles

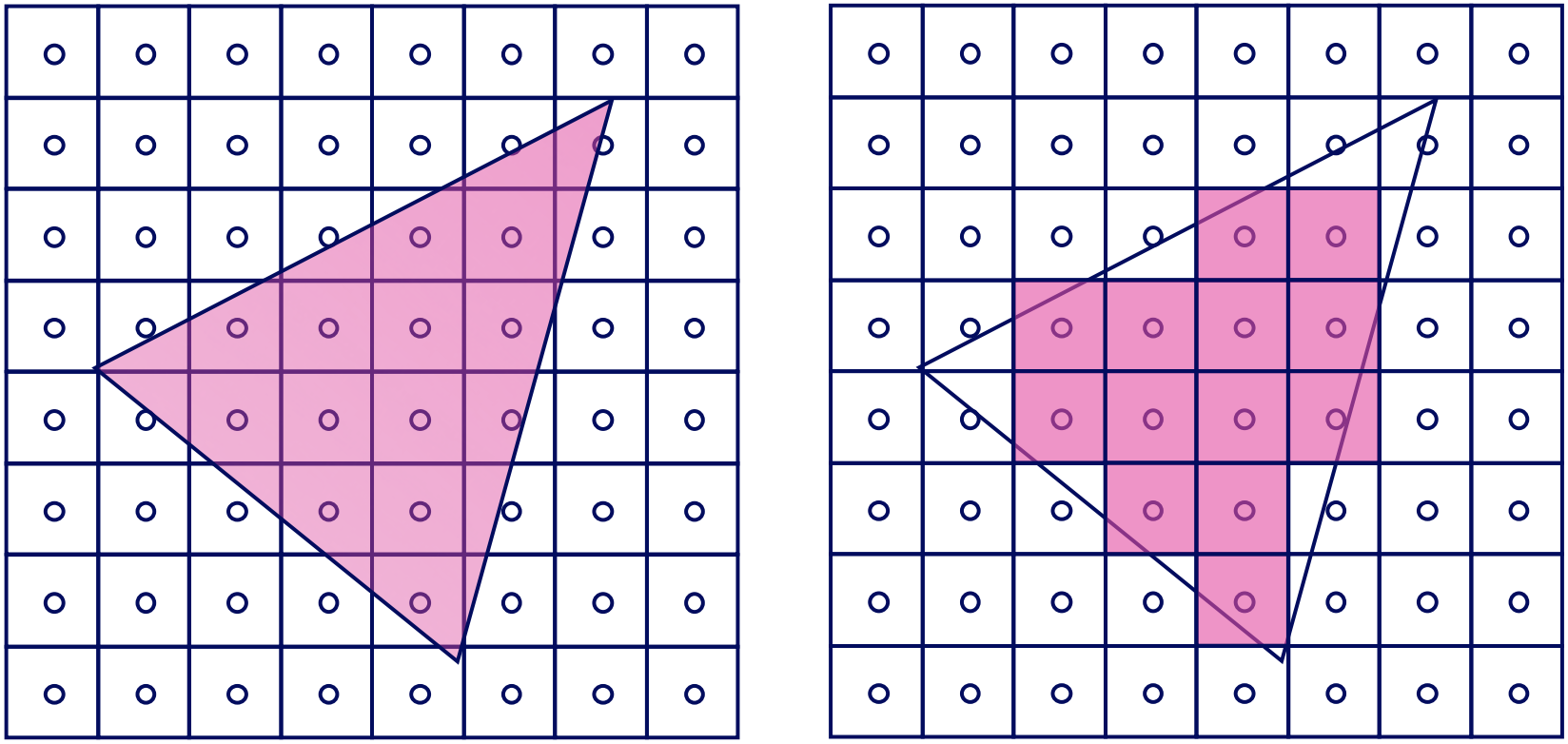
- why?

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





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



Modern Rasterization

Define a triangle as follows:

Scaled Implicit Line Equation

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;
      }
}
```

Edge Equations: Code

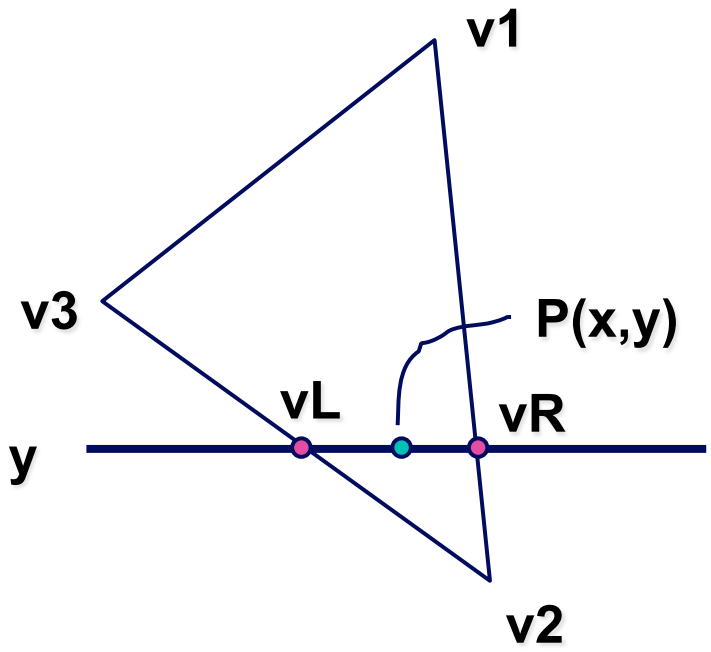
```
  // more efficient inner loop
for (int y = yMin; y <= yMax; y++) {
      float e0 = a0*xMin + b0*y + c0;
      float e1 = a1*xMin + b1*y + c1;
      float e2 = a2*xMin + b2*y + c2;
      for (int x = xMin; x <= xMax; x++) {
        if (e0 > 0 && e1 > 0 && e2 > 0)
          Image[x][y] = TriangleColor;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

1. Bilinear Interpolation

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



2. Plane Equation

- $v = Ax + By + C$

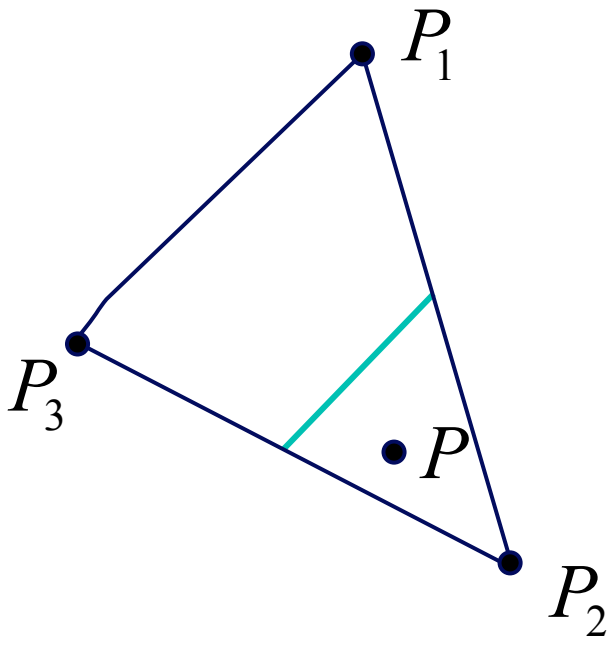
3. Barycentric Coordinates

- ***weighted combination of vertices***

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

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

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



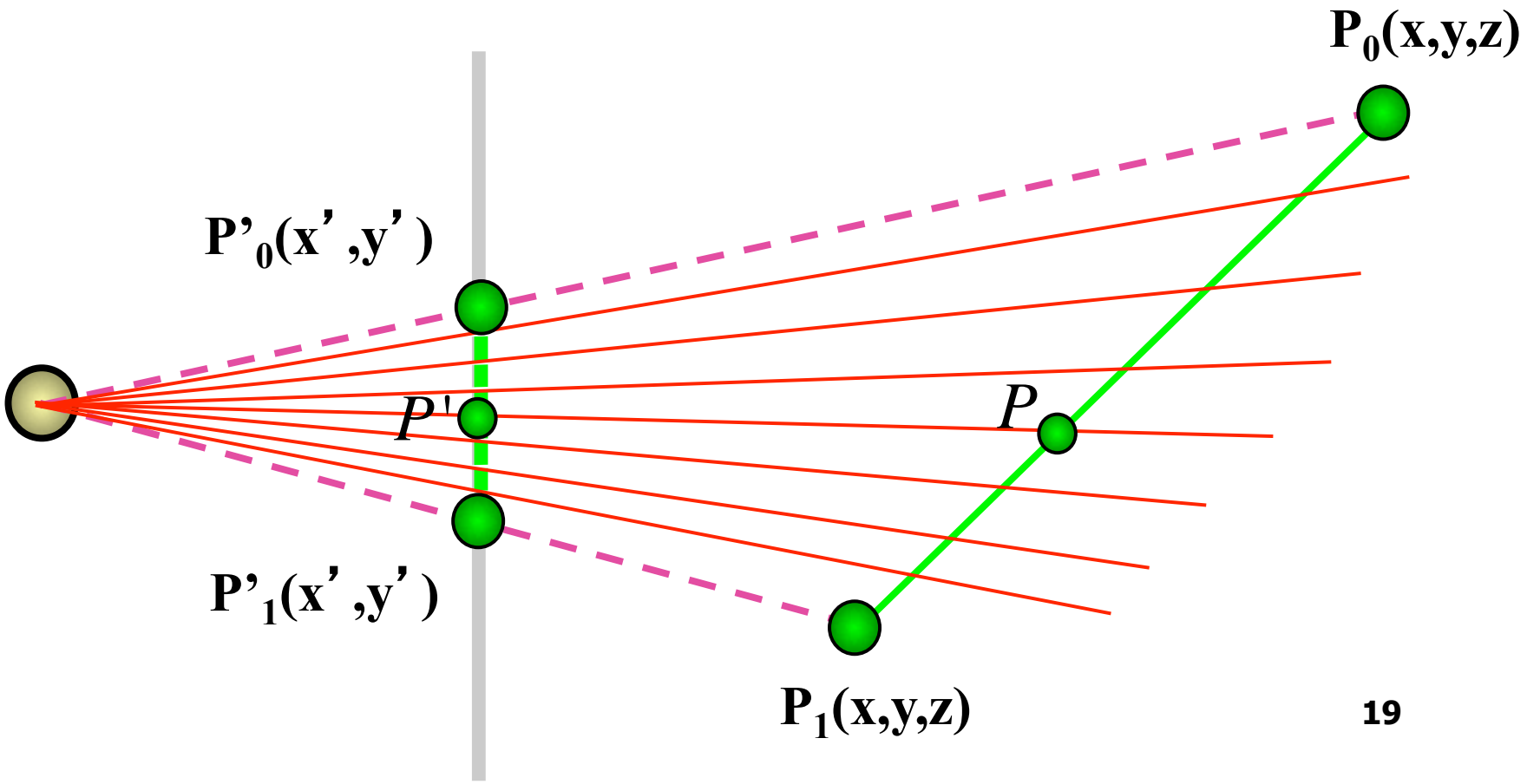
Barycentric Coordinates

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

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

- computing Barycentric coordinates

Interpolation: Screen vs World Space



Perspective-correct interpolation

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

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