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

Scan Conversion:
The process of identifying all the pixels that lie within a given triangle.
Implicit, Explicit, and Parametric equations for defining geometry

1. Implicit
   \[ F(x,y) > 0 \]
   \[ F(x,y) < 0 \]

2. Explicit
   \[ y = mx + b \]

3. Parametric
   Point \( P(t) \) is a function of an underlying parameter, \( t \). Useful to think of \( t \) as being time.
Lines and Curves

**Explicit**

- line
  \[ y = mx + b \]
  \[ y = y_1 + \Delta y = y_1 + m\Delta x \]
  \[ y = y_1 + \frac{(y_2 - y_1)(x - x_1)}{(x_2 - x_1)} \]
- circle
  \[ y = \pm \sqrt{r^2 - x^2} \]
- plane
  \[ z = Ax + By + C \]
- sphere
  \[ z = \pm \sqrt{r^2 - x^2 - y^2} \]
Lines and Curves

**Parametric**

- **line**
  \[ \mathbf{p}(t) = \mathbf{p}_1 + t (\mathbf{p}_2 - \mathbf{p}_1) = (1-s)\mathbf{p}_1 + s\mathbf{p}_2 \]

- **circle**
  \[ x(t) = r \cos(t) + \mathbf{p}_0 \]
  \[ y(t) = r \sin(t) + \mathbf{p}_0 \]
  \[ t \in [0, 2\pi] \]

- **plane**
  \[ \mathbf{p}(s,t) = \mathbf{p}_0 + s (\mathbf{p}_1 - \mathbf{p}_0) + t (\mathbf{p}_2 - \mathbf{p}_0) \]
  \[ \begin{bmatrix} x(s,t) \\ y(s,t) \\ z(s,t) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + s \begin{bmatrix} x_1-x_0 \\ y_1-y_0 \\ z_1-z_0 \end{bmatrix} + t \begin{bmatrix} x_2-x_0 \\ y_2-y_0 \\ z_2-z_0 \end{bmatrix} \]
Lines and Curves

**Implicit**

2D

\[ F(x_1, y_1) = 0 \]

3D

\[ F(x, y, z) = 0 \]

**Line**

\[ y = y_1 + \Delta y = y_1 + m \Delta x \]

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

**Circle**

\[ r^2 = x^2 + y^2 \]

\[ 0 = x^2 + y^2 - r^2 \]

\[ F(x, y) = 0 \text{ on circle} \]

\[ F(x, y) < 0 \text{ inside} \]

\[ F(x, y) > 0 \text{ outside} \]

**3D Point**

\[ 0 = x(y_2 - y_1) + y(x_2 - x_1) + y_1x_2 - y_2x_1 \]
Polygons

Interactive graphics uses polygons

Simple: edges do not self-intersect
Convex: interior angles $\theta_i \leq 180^\circ$

More generally, 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*
What is Scan Conversion? (a.k.a. Rasterization)

Set all pixels/fragments whose center point is "inside"
Modern Rasterization

Define a triangle as follows:
Scaled Implicit Line Equation

From before: \( 0 = x(y_2 - y_1) + y(x_1 - x_2) + y_1x_2 - y_2x_1 \)

\[ F(x_i, y_i) \quad 0 = Ax + By + C \]

Now develop \( F(x_i, y_i) \) such that \( F'(x_3, y_3) = +1 \)

Define \( K = F(x_3, y_3) \)

Then \( F'(x_i, y_i) = \frac{F(x_i, y_i)}{K} \)

i.e., \( F'(x_i, y_i) = \frac{A}{K}x + \frac{B}{K}y + \frac{C}{K} \)
Edge Equations: Code

```c
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;
    }
}
Triangle Rasterization Issues

What about pixels exactly on the edge?

Partial solutions: anti-aliasing; set a pixel "partly on" based on the fraction of coverage.

Choices:
1. Draw them both. Result depends on order.
2. Don't draw them. × gap.
3. Use a consistent but arbitrary rule. E.g.: draw pixels on left or top boundaries.