Scan Conversion - Rasterization

- Convert continuous rendering primitives into discrete fragments/pixels
  - Lines
    - Bresenham
  - Triangles
  - Flood Fill
  - Scanline
  - Implicit formulation

Lines and Curves

- Explicit - one coordinate as function of the others
  - \( y = f(x) \)
  - \( z = f(x, y) \)
  - Line: \( y = mx + b \)
  - \( y = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1) + y_1 \)
  - Circle: \( y = \pm \sqrt{r^2 - x^2} \)

- Implicit - define as “zero set” of function of all the parameters
  - \( \{(x, y): F(x, y) = 0\} \)
  - \( \{(x, y, z): F(x, y, z) = 0\} \)
  - Defines partition of space
    - \( \{(x, y): F(x, y) > 0\} \)
    - \( \{(x, y): F(x, y) = 0\} \)
    - \( \{(x, y): F(x, y) < 0\} \)

- Parametric – all coordinates as functions of common parameter
  - \( (x, y) = (f_1(t), f_2(t)) \)
  - \( (x, y, z) = (f_1(t), f_2(t), f_3(t)) \)
  - Line: \( x(t) = x_0 + t(x_1 - x_0), \quad y(t) = y_0 + t(y_1 - y_0) \)
  - Circle: \( x(\theta) = r \cos(\theta), \quad y(\theta) = r \sin(\theta) \)
  - \( \theta \in [0, 2\pi] \)

Basic Line Drawing

Assume \( x_1 < x_2 \) & line slope absolute value is \( \leq 1 \)

- Line: \( y = f(x) \)
  - \( \text{Int} \): \( f(x) = \text{int}(f(x)) \)
  - \( \text{PlotPixel}(x, f(x)) \)
  - \( \text{Floor}(x) \) for \( x \) outside \( x_1 \) to \( x_2 \)

Questions:
- Can this algorithm use integer arithmetic?
- Does it accumulate error?
- Is the error significant?
Recursive Line Drawing

Simple, recursive, integer, line drawing:

```c
Line (x0, y0, x1, y1)
begin
  int x, y;
  x = (x0 + x1) / 2;
  y = (y0 + y1) / 2;
  if (x == x0 and y == y0) or
     (x == x1 and y == y1)
    return ;
else begin
  PrintPixel (x, y) ;
  Line (x0, y0, x, y) ;
  Line (x, y, x1, y1) ;
end ;
end ;
```

Questions:
- Does the algorithm accumulate error?
- Is it significant?

More Problems:
- Line not drawn sequentially
- Function call for each pixel drawn

We want a faster algorithm!

Midpoint (Bresenham) Algorithm

• Assumptions:
  \[ x_2 > x_1, y_2 > y_1 \text{ and } \frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1} < 1 \]

• Idea:
  - Define error function
  - Proceed along the line incrementally
  - Select direction that minimizes accumulated error

Definitions

\[ e = \{(x, y) | ax + by + c = xady - ydx + c = 0\} \]

\[ d(x, y) = 2(xady - ydx + c) \]

Bresenham Algorithm

- Given point \( P = (x, y) \) \( d(x, y) \) is signed distance of \( P \) to \( \tau \) (up to normalization factor)
- \( d \) is zero for \( P \in \tau \)
  \( \Rightarrow d \) may serve as error function to be minimized
- Starting point satisfies \( d(x_0, y_0) = 0 \)
- Each step moves right (east) or upper right (northeast)

Midpoint Line Drawing (cont’d)

• Sign of \( d(x_1 + 1, y_1 + \frac{1}{2}) \) indicates if to move east or northeast
  - At \((x, y)\)
  \[ d_{\text{start}} = d(x_1 + 1, y_1 + \frac{1}{2}) = 2dy - dx \]

• Increment in \( d \) (after each step)
  - Move east \( \Delta_e = d(x + 2, y + \frac{1}{2}) - d(x + 1, y + \frac{1}{2}) = 2(x + 2)(dy - (y + \frac{1}{2})(dx+c)) - 2(x + 1)(dy - (y + \frac{1}{2})(dx+c) = 2dy \)
  - Move northeast \( \Delta_{en} = d(x_2 + 1, y_2 + \frac{1}{2}) - d(x_1 + 1, y_1 + \frac{1}{2}) = 2(x + 2dy - (y + \frac{1}{2})(dx+c)) - 2(x + 1)(dy - (y + \frac{1}{2})(dx+c) = 2(dy - dx) \)
Midpoint Examples

- Question: Is there a problem with this algorithm (horizontal vs. diagonal lines)?
- Comment: extends to higher order curves – e.g. circles

Error Function Intuition

- Error function $d$ can be viewed as explicit surface:

$$d(x,y) = 2(x dy - y dx + c)$$