Chapter 7

Scan Conversion - Drawing on Raster Display (part 1 - Lines)

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

Scan Conversion - Lines
- Given segment equation fill in the pixels
- In drawings below - grid points = centers of pixels

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

Lines and Curves
- Parametric - all coordinates as functions of common parameter
  - line: \( (x, y) = (f(t), f_z(t)) \)
  - \( (x, y, z) = (f_x(u, v), f_y(u, v), f_z(u, v)) \)
  - \( x(t) = x_1 + f(x_2 - x_1) \)
  - \( y(t) = y_1 + f(y_2 - y_1) \)
  - \( t \in [0, 1] \)
  - \( x(\theta) = r \cos(\theta) \)
  - \( y(\theta) = r \sin(\theta) \)
  - \( \theta \in [0, 2\pi] \)
**Computer Graphics**

### Lines and Curves
- Implicit - define as "zero set" of function of all the parameters
  \[(x,y): F(x,y) = 0\]
  \[\{(x,y) : F(x,y) = 0\}\]
  - Defines partition of space
  \[\{(x,y) : F(x,y) > 0\}, \{(x,y) : F(x,y) = 0\}, \{(x,y) : F(x,y) < 0\}\]

### Basic Line Drawing
Assume \(x < x₁ \& \) line slope absolute value is \(\leq 1\)

#### Line
- From \((xₘ, yₘ), (xₙ, yₙ)\)
- \(d_x, dx, dx + xₙ - xₘ\)
- \(d_y, dy, dy + yₙ - yₘ\)
- \(\text{slope} = \frac{dy}{dx}\)
- \(y = yₘ\)
- For \(x\) from \(xₘ\) to \(xₙ\), do
  ```
  begin
  while \(x < xₙ\) do
    RoundPlotPixel \((x, y)\)
    \(x = x + \text{round}()\)
  end
  ```

#### Questions:
Can this algorithm use integer arithmetic?

### Midpoint (Bresenham) Algorithm
- **Assumptions:**
  - \(x₂ > x₁, y₂ > y₁\) and \(\frac{dy}{dx} = \frac{y₂ - y₁}{x₂ - x₁} < 1\)
- **Idea:**
  - Proceed along the line incrementally
  - Have ONLY 2 choices
  - Select one that minimizes error (distance to line)

#### Distance (error):
\[
\tau = \{(x,y) | ax + by + c = xdy - ydx + c = 0\}
\]
\[
d(x,y) = 2(axdy - bydx + c)
\]
- Given point \(P = (x,y)\) of \(\tau\) is signed distance of \(P\) to \(\tau\) (up to normalization factor)
- \(d\) is zero for \(P \in \tau\)

### Midpoint Line Drawing (cont'd)
- **Starting point satisfies** \(d(x₁, y₁) = 0\)
- Each step moves right (east) or upper right (northeast)
- **Sign of** \(d(x + 1, y + \frac{1}{2})\) indicates if to move east or northeast
Midpoint Line Algorithm

Top (x1, y1, x2, y2)

Main:

- if y1 < y2:
  - plot y1
  - change x:
    - if diff > 0:
      - x = x + 1
    - else:
      - x = x + 1
  - change y:
    - y = y + 1
- else:
  - plot x1
  - change y:
    - if diff > 0:
      - y = y + 1
    - else:
      - y = y + 1
  - change x:
    - x = x + 1

Increment d (after each step):
- if move east: \( \Delta_e = d(x + 1, y) = d(x + 1, y + 1) \)
- if move northeast: \( \Delta_{ne} = d(x + 1, y + 1) = d(x + 1, y + 1) \)

Midpoint Line Algorithm (version 1)

Main:

- if y1 < y2:
  - plot y1
  - change x:
    - if diff > 0:
      - x = x + 1
    - else:
      - x = x + 1
  - change y:
    - y = y + 1
- else:
  - plot x1
  - change y:
    - if diff > 0:
      - y = y + 1
    - else:
      - y = y + 1
  - change x:
    - x = x + 1

Increment d (after each step):
- if move east: \( \Delta_e = d(x + 1, y) = d(x + 1, y + 1) \)
- if move northeast: \( \Delta_{ne} = d(x + 1, y + 1) = d(x + 1, y + 1) \)

Midpoint Line Drawing (cont’d)

- Insanely efficient version (less computations inside the loop)
- compute d incrementally
- At \((x_1, y_1)\):
  \[ d_{\text{start}} = d(x_1 + 1, y_1 + \frac{1}{2}) = 2 dy - dx \]
- Increment in d (after each step):
  - if move east: \( \Delta_e = d(x + 1, y) = d(x + 1, y + 1) \)
  - if move northeast: \( \Delta_{ne} = d(x + 1, y + 1) = d(x + 1, y + 1) \)

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

Copyright 2005, Alla Sheffer