Chapter 4

Scan Conversion – Drawing Lines on Raster Display

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**
  
  \[ y = \pm \sqrt{r^2 - x^2} \]

---

Lines and Curves

- 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**
  
  \[x(t) = x_0 + t(x_1 - x_0)\]
  
  \[y(t) = y_0 + t(y_1 - y_0)\]
  
  \[t \in [0,1]\]

  **circle**
  
  \[x(\theta) = r \cos(\theta)\]
  
  \[y(\theta) = r \sin(\theta)\]
  
  \[\theta \in [0,2\pi]\]
Lines and Curves

- 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
  \( \{(x, y): F(x, y) > 0\}, \{(x, y): F(x, y) = 0\}, \{(x, y): F(x, y) < 0\} \)

\[
\begin{align*}
F(x, y) &= (x - x_0)dy - (y - y_0)dx \\
F(x, y) &= 0 \\
F(x, y) &> 0 \\
F(x, y) &< 0
\end{align*}
\]

\[
\begin{align*}
F(x, y) &= x^2 + y^2 - r^2 \\
(x, y) \text{ is on line} & \quad F(x, y) = 0 \\
(x, y) \text{ is below line} & \quad F(x, y) > 0 \\
(x, y) \text{ is above line} & \quad F(x, y) < 0 \\
(x, y) \text{ is on circle} & \quad (x, y) \text{ is outside} \\
(x, y) \text{ is inside}
\end{align*}
\]

Basic Line Drawing

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

```
Line( x1, y1, x2, y2 )
begin
\( \text{if \( dx = 0 \)} \)
\( \text{if \( \text{slope} = 0 \)} \)
\( \text{if \( \text{slope} = \infty \)} \)
\( \text{if \( \text{slope} = -\infty \)} \)
\( \text{do \( \text{PlotPixel( } x, \text{Round( } y \text{ ) } ) \);} \)
\( y \leftarrow y + \text{slope} \)
end
end
```

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 (x1, y1, x2, y2)
begin
    int x, y;
    x = (x1 + x2) / 2;
    y = (y1 + y2) / 2;
    if ((x = x1 and y = y1) or
        (x = x2 and y = y2))
        return;
    else begin
        PlotPixel (x, y);
        Line (x1, y1, x, y);
        Line (x, y, x2, y2);
        end;
end;
```

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

Recursive Line Drawing (cont’d)

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**

\[
\begin{align*}
\tau &= \{(x, y) | ax + by + c = xdy - ydx + c = 0 \} \\
d(x, y) &= 2(xdy - ydx + c)
\end{align*}
\]

Bresenham Algorithm

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

- 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 \(d(x_1, y_1) = 0\)

- 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 $(x_1, y_1)$
  \[ d_{start} = d(x_1 + 1, y_1 + \frac{1}{2}) = 2dy - dx \]
- 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 $\Delta_{ne} = d(x_1 + 2, y_1 + \frac{3}{2}) - d(x_1 + 1, y_1 + \frac{1}{2}) = 2((x + 2)dy - (y + \frac{3}{2})dx + c)) - 2((x + 1)dy - (y + \frac{1}{2})dx + c) = 2(dy - dx)$

Midpoint Line Algorithm

```c
Line (x1, y1, x2, y2)
begin
  int x, y, dx, dy, d, d', d', d_{ne} = 2 * (dy - dx);
  PlotPixel (x, y); // Plot pixel at (x, y)
  while (x < x2) do
    if (d < 0) then
      begin
        d := d + d_{ne};
        x := x + 1;
      end ;
    else begin
      d := d + d_{ne};
      x := x + 1;
      y := y + 1;
    end ;
  PlotPixel (x, y); // Plot pixel at (x, y)
end ;
```

Copyright 2004, Alla Sheffer
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:
  \[ d(x,y) = 2(xdy-ydx+c) \]