Scan Conversion: Drawing Lines on Raster Display

Raster Display
- Discrete grid of elements (frame buffers/pixels)
- Shapes drawn by setting the "right" elements
- Frame buffer is scanned, one line at a time, to refresh the image (as opposed to vector display)
- Properties:
  - Difficult to draw smooth lines
  - Displays only a discrete approximation of any shape
  - Refresh of entire frame buffer

Terminology
- Pixel: Picture element
  - Smallest accessible element in picture
  - Usually rectangular or circular
- Aspect Ratio: Ratio between physical dimensions of pixel
  - (not necessarily 1:1)
- Dynamic Range: Ratio between minimal (not zero) and maximal light intensity emitted by displayed pixel

Basic Line Drawing
Assume \( x_1 < x_2 \) and line slope absolute value is \( \leq 1 \)

```
Line (x1, y1, x2, y2)
begin
  dx = x2 - x1,
dy = y2 - y1;
slope = dy/dx;
y = y1;
for x from x1 to x2 do
  PlotPixel (x, Round(y));
y = y + slope;
end;
```

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

Recursive Line Drawing
Simple, recursive, integer, line drawing:
```
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;
```

Questions:
- How does the algorithm work?
- Does the algorithm accumulate error?
- Is it significant?
Recursive Line Drawing (cont’d)

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

We want a faster and accurate algorithm!

Midpoint (Bresenham) Algorithm (cont’d)

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

- Given point \( P(x, y) \), \( d(x, y) \) is the signed distance of \( P \) to \( r \) (up to normalization factor)
- \( d \) is zero for \( P \notin r \)
  - \( d \) 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 \( d(x_1+1, y_1+1/2) \) indicates whether to move east or northeast
- At \((x_0, y_0)\)
  \[ d_{\text{incr}} = d(x_1+1, y_1+1/2) = 2dy - dx \]
- Increment in \( d \) (after each step)
  - Move east
    \[ \Delta_x = d(x + 2, y + 1/2) - d(x + 1, y + 1/2) = 2(2x + 2dy - (y + 1/2)adx + c) - 2(x + 1)dy - (y + 1/2)adx + c = 2dy \]
  - Move northeast
    \[ \Delta_x = d(x + 2, y + 3/2) - d(x + 1, y + 1/2) = 2(2x + 2dy - (y + 3/2)adx + c) - 2(x + 1)dy - (y + 1/2)adx + c = 2(2dy - dx) \]

Midpoint Line Algorithm

```
Line (x0, y0, x1, y1):
begin
    \( x = x_0 \), \( y = y_0 \), \( d = \Delta_y \), \( \Delta_x + \Delta_y \);
    \( d = 2 \times d \);
    \( \Delta_x = 2 \times (y_1 - y_0) \);
    call PlotPixel(x, y);
    while \( x < x_1 \) do
        if \( d < 0 \) then begin
            \( x = x + 1 \);
            \( x = x + 1 \);
            \( x = x + 1 \);
            \( x = x + 1 \);
        end else begin
            \( x = x + 1 \);
            \( x = x + 1 \);
            \( x = x + 1 \);
            \( x = x + 1 \);
        end;
        call PlotPixel(x, y);
end;
```

Midpoint Examples

- Midpoint (squares) vs. Recursive (squares) line drawing
- Midpoint line drawing

Question: Is there a problem with the Midpoint algorithm (hint: horizontal vs. diagonal lines)?

Comment: Algorithm extends to higher order curves – e.g. circles
Circle Midpoint

- For first quadrant, start from \((x_0, y_0) = (0, R)\)
- Move east or south-east
- \(d(x, y)\) will be a threshold criteria at the midpoint

Circle Octant Examples

- Error Function Intuition
  - Error function \(d\) can be viewed as explicit surface:
    \[
d(x, y) = 2(x_0y_0 - d(c + c))
    \]
  - Implicit equation of origin-centered circle \(C\) is
    \[
x^2 + y^2 - R^2 = 0
    \]
  - Error functional
    \[
d(x, y) = x^2 + y^2 - R^2
    \]
  - This last change has no effect on threshold criteria
  - \(\Delta_x\) and \(\Delta_y\) not constant anymore

Midpoint Circle Algorithm

```
CircleOctant2(R)
begin
int x, y, d, R;
while (y > x) do
if (d < 0) then
begin
  d := d + 2x + 3;
  x := x + 1;
end;
else begin
  d := d + 2(R-x-y) + 5;
  y := y + 1;
end;
end;
end;
```

Question: What will happen if we change “while \((y > x)\)” to “while \((y > 0)\)?”

Computer Graphics
Scan Conversion

Copyright 1996 C. Gotsman & G. Elber
Computer Science Dept. Technion