Circle Generation

Outline
- Circle generation problem
- Circle equations
- Naïve approach:
  - Based on parametric representation
  - Midpoint circle drawing algorithm
  - Ellipse generation problem

Circle Generation Problem
- How to display a circle on a monitor?
  - Select pixels that are closest to the circle
  - Do it efficiently
- Simplifications:
  - Circles are 1 pixel wide
  - No need for anti-aliasing

Circle Equations
- To specify a circle:
  - Radius : r
  - Center position : (x₀, y₀)
- Parametric representation:
  - y(θ) = y₀ + r * cos(θ)
  - x(θ) = x₀ + r * sin(θ)
- Implicit representation:
  - F(x,y) = (x - x₀)² + (y - y₀)² - r² = 0

Approach Based on Parametric Representation
- Algorithm:
  - for (float θ=0 ; θ<2π ; θ+=dθ) {
    int y = round(r * cos(θ));
    int x = round(r * sin(θ));
    draw(x₀+x, y₀+y);
  }
- Computational cost:
  - O(1/dθ)

Eight-Way Symmetry
- Only need to calculate 1/8 circle.
  - From (0,0) to (b,b)
  - Corresponding pixels of (x,y):
    - (y, x)
    - (y, -x)
    - (x, -y)
    - (-x, -y)
    - (-y, x)
    - (-y, -x)
    - (-x, y)
    - (-x, -y)
Approach using Symmetry

- Algorithm:
  - for (float θ=0 ; θ<=π/4 ; θ+=dθ) {
    - int y = round(r * cos(θ));
    - int x = round(r * sin(θ));
    - draw(x0+x, y0+y);
    - draw(x0+y, y0+x);
    - draw(x0+y, y0-x);
    - ...;
  }
- Computational cost:
  - About 1/8 of original approach.

Limitations

- How to step through θ?
  - Small spacing -> inefficiency
  - Large spacing -> scatter points
  - The proper value of dθ depends on the size (radius) of the circle
- High computational cost
  - Both sine & cosine functions require Taylor Expansion
  - High computational cost

Midpoint Approach

- The relative position of point (u,v) to the circle can be determined using the sign of the implicit circle function:
  - On the circle
    - F(u,v)=0
  - Outside the circle
    - F(u,v)>0
  - Inside the circle
    - F(u,v)<0

Initial Condition

- Curve starts from (0,r)
  - (0,r) should be picked.
  - M is (1,r−½)
- Set d=F(M)
  - (u+1, v) is picked
    - M’ is (u+2,v−½)
    - d’=F(M’)
      =((u+2)² + (v−½)²) - r²
    - Previous d value is known
      - d=F(M)
        =((u+1)² + (v−½)²) - r²
    - Difference: d’−d = 2u+3
    - Update: d’ = d + 2u+3
  - (u+1,v−1) is picked
    - M’ is (u+2,v−1½)
    - d’=F(M’)
      =((u+2)² + (v−1½)²) - r²
    - Previous d value:
      - d=F(M)
        =((u+1)² + (v−½)²) - r²
    - Difference:
      - d’−d = 2u-2v+5
    - Update:
      - d’ = d + 2u-2v+5

Next Decision (Scenario 1)

Next Decision (Scenario 2)
Overall Algorithm

- int x=0, y=r;
- float d = 5.0 / 4 - r;
- while (y >= x) {
  - draw(x, y);
  - if (d < 0) {
    - d += x * 2 + 3;
    - x ++;
  } else {
    - d += (x - y) * 2 + 5;
    - x ++; y --;
  }
- }

Program Transformation

- int x=0, y=r;
- int d = 1 - r;
- while (y >= x) {
  - draw(x, y);
  - if (d < 0) {
    - d += x * 2 + 3;
    - x ++;
  } else {
    - d += (x - y) * 2 + 5;
    - x ++; y --;
  }
- }

How about Ellipse

- Implicit representation:
  - \( F(x, y) = x^2/a^2 + y^2/b^2 - 1 = 0 \)
- Eight-way symmetry does not exist.
  - Four-way symmetry can be used.
  - Need to draw 1/4 of the curve, instead of 1/8.

Two Different Procedures Needed

- Single procedure cannot handle the full ¼ curve
  - Split curve into 2 regions at location where slope \(|m| = 1\)
  - \(|m| < 1\) in region 1:
    - Increase x by 1 each time
  - \(|m| > 1\) in region 2:
    - Increase y by 1 each time