Circle Generation

Outline
- Circle generation problem
- Straightforward approach
- Midpoint approach
- 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 Equation
- To specify a circle:
  - Radius: \( r \)
  - Center position: \((x_0, y_0)\)
- Parametric representation:
  - \( y(\theta) = y_0 + r \cdot \cos(\theta) \)
  - \( x(\theta) = x_0 + r \cdot \sin(\theta) \)
- Implicit representation:
  - \( F(x, y) = (x - x_0)^2 + (y - y_0)^2 - r^2 = 0 \)

Straightforward Approach
- Based on parametric representation
- Algorithm:
  - for (float \( \theta = 0; \theta < 2\pi; \theta += \Delta \theta \) ) {
    - int \( y = \text{round}(r \cdot \cos(\theta)) \);
    - int \( x = \text{round}(r \cdot \sin(\theta)) \);
    - draw(x0+x, y0+y);
  }
- Computational cost:
  - \( O(1/\Delta \theta) \)

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

Approach using Symmetry

• Algorithm:
  • for (float $\theta = 0$; $\theta < \pi/4$; $\theta += d\theta$) {
    • int $y = \text{round}(r * \cos(\theta))$;
    • int $x = \text{round}(r * \sin(\theta))$;
    • draw($x_0+x$, $y_0+y$);
    • draw($x_0+y$, $y_0+x$);
    • draw($x_0+y$, $y_0-x$);
    • ...
  }
• Computational cost:
  • About 1/8 of original approach.

Problems of the Straightforward Approach

• How to step through $\theta$?
  • Small spacing -> inefficiency
  • Large spacing -> scatter points
  • The proper value of $d\theta$ depends on the size (radius) of the circle
• High computational cost
  • Both sine and cosine function require lots of computation

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 $(1,r-\frac{1}{2})$
• Set $d=F(M)$
  • $=1 + (r-\frac{1}{2})^2 - r^2$
  • $=1 + (r^2-r+\frac{1}{4}) - r^2$
  • $=1\frac{1}{2} - r$

Next Decision (Scenario 1)

• If $(u+1,v)$ is picked
  • $M'$ is $(u+2,v-\frac{1}{2})$
  • Current $d=F(M)$
    • $=(u+1)^2 + (v-\frac{1}{2})^2 - r^2$
  • Next $d'=F(M')$
    • $=(u+2)^2 + (v-\frac{1}{2})^2 - r^2$
  • Difference:
    • $d' - d = 2u+3$
  • Update:
    • $d' = d + 2u+3$

Next Decision (Scenario 2)

• If $(u+1,v-1)$ is picked
  • $M'$ is $(u+2,v-1\frac{1}{2})$
  • Current $d=F(M)$
    • $=(u+1)^2 + (v-1\frac{1}{2})^2 - r^2$
  • Next $d'=F(M')$
    • $=(u+2)^2 + (v-1\frac{1}{2})^2 - r^2$
  • Difference:
    • $d' - d = 2u-2v+5$
  • Update:
    • $d' = d + 2u-2v+5$
Overall Algorithm
- int x=0, y=r;
- float d = 5.0 / 4 – r;
- while (y >= x) {
  - draw(x0+x, y0+y);
  - draw(x0+y, y0+x);
  - 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(x0+x, y0+y);
  - draw(x0+y, y0+x);
  - 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 1/4 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