Scan Converting Circles

\[(x - x_0)^2 + (y - y_0)^2 = R^2\]

- Symmetry: If \((x_0 + a, y_0 + b)\) is on the circle, so are \((x_0 \pm a, y_0 \pm b)\) and \((x_0 \pm b, y_0 \pm a)\), hence there’s an 8-way symmetry.

- But in a practical setting of considering pixel values, it depends on the fact that \(x_0\) and \(y_0\) are integers.
Using the Symmetry

- We will scan top right 1/8 of circle of radius R

- It starts at \((x_0, y_0 + R)\)
Sketch of Algorithm

\[
y = y_0 + R; \ x = x_0; \ \text{Pixel}(x, \ y);
\]
\[
\text{for} \ (x = x_0 + 1; \ (x - x_0) < (y - y_0); \ x++) \{
\]
\[
\text{if} \ (\text{decision\_var} < 0) \{
\]
\[
\text{/* move east*/}
\]
\[
\text{update decision\_var;}
\]
\[
\}
\]
\[
\text{else} \{
\]
\[
\text{/* move south east*/}
\]
\[
\text{update decision\_var;}
\]
\[
y--;\}
\]
\[
\text{Pixel}(x, \ y);
\]
\[
\}
\]

- Note: can replace all occurrences of \(x_0\) and \(y_0\) with 0,0 and \(\text{Pixel}(x,y)\) with \(\text{Pixel}(x_0 + x, y_0 + y)\).

- Essentially a shift of coordinates
Bad Alternatives

• Version 1:
  For \( x = 0 \) to 360
  Pixel (round \((R \times \cos(x))\), round\((R \times \sin(x))\));

• Version 2
  For \( x = -R \) to \( R \)
  \( y = \sqrt{R \times R - x \times x} \);
  Pixel (round\((x)\), round\((y)\));
  Pixel (round\((x)\), round\(( -y)\));

• HW: explain why these are bad.
What we need for incremental algorithm

• Need a decision variable, i.e., something that is negative if we should move E, position if we should move SE (or vice versa).

• Follow line strategy: Use the implicit equation of circle

\[ h(x,y) = x^2 + y^2 - R^2 = 0 \]

\( h(x,y) \) is zero on the circle, negative inside it, positive outside.

• If we are at pixel \( (x,y) \), examine \( (x + 1, y) \) and \( (x + 1, y - 1) \).

• Again compute \( h \) as the midpoint = \( h(\text{midpoint}) \)
Decision Variable

- Evaluate \( h(x,y) = x^2 + y^2 - R^2 \)

  at the point \( \left( x + 1, y - \frac{1}{2} \right) \)

- What we are asking is this: “Is

  \[
  h\left( x + 1, y - \frac{1}{2} \right) = (x + 1)^2 + \left( y - \frac{1}{2} \right)^2 - R^2
  \]

  positive or negative?”

- If it is negative there, this midpoint is inside the circle, so the *vertical* distance to the circle is less at \( (x + 1, y) \) than at \( (x + 1, y-1) \).

- If it is positive, the opposite is true.
Is this the right decision variable?

- It makes our decision based on vertical distance.
- For lines, that was ok, since $d$ and $d_{\text{vert}}$ were proportional.
- For circles, no longer true:

$$d(((x + 1, y), \text{Circ}) = \sqrt{(x + 1)^2 + y^2} - R$$

$$d(((x + 1, y - 1), \text{Circ}) = \sqrt{(x + 1)^2 + (y - 1)^2} - R$$

- We ask which $d$ is closer to zero, i.e., which of the two values below is closer to $R$:

$$\sqrt{(x + 1)^2 + y^2} \quad \text{or} \quad \sqrt{(x + 1)^2 + (y - 1)^2}$$
Alternate Phrasing (1/3)

- We could ask instead

\[(*) \text{“Is } (x + 1)^2 + y^2 \text{ or } (x + 1)^2 + (y - 1)^2 \text{ closer to } R^2?”\]

- The two values in equation (*) above differ by

\[
[(x + 1)^2 + y^2] - [(x + 1)^2 + (y - 1)^2] = 2y - 1
\]
Alternate Phrasing (2/3)

- So the second value, which is always the lesser

is closer if its difference from R2 is less

\[
\text{than } \left(\frac{1}{2}\right)(2y-1) \quad \text{i.e., if}
\]

\[
R^2 - [(x+1)^2 + (y-1)^2] < \frac{1}{2}(2y-1)
\]

then

\[
0 < y - \frac{1}{2} + (x+1)^2 + (y-1)^2 - R^2
\]

so

\[
0 < (x+1)^2 + y^2 - 2y + 1 + y - \frac{1}{2} - R^2
\]

so

\[
0 < (x+1)^2 + y^2 - y + \frac{1}{2} - R^2
\]

so

\[
0 < (x+1)^2 + \left(y - \frac{1}{2}\right)^2 + \frac{1}{4} - R^2
\]
Alternate Phrasing (3/3)

- So the radial distance decision is whether

\[ d1 = (x + 1)^2 + \left(y - \frac{1}{2}\right)^2 + \frac{1}{4} - R^2 \]

is positive or negative,

- And the vertical distance decision is whether

\[ d2 = (x + 1)^2 + \left(y - \frac{1}{2}\right)^2 - R^2 \]

is positive or negative; \(d1\) and \(d2\) are \(\frac{1}{4}\) apart.

- The integer \(d2\) is positive only if \(d2 + \frac{1}{4}\) is positive (except special case where \(d2 = 0\)).

Hence, aside from ambiguous cases, the two are the same.
Incremental Computation, Again (1/2)

• How should we compute the value of

\[ h(x, y) = (x + 1)^2 + \left(y - \frac{1}{2}\right)^2 - R^2 \]

at successive points?

• Answer: Note that

\[ h(x + 1, y) - h(x,y) \]

is just

\[ 2x + 3 = \Delta_E(x,y) \]

and that

\[ h(x + 1, y-1) - h(x,y) \]

is just

\[ \Delta_{SE}(x,y) = 2x + 3 - 2y + 2 \]
Incremental Computation (2/2)

- So if we move east, update by adding $2x + 3$

- And if we move SE, update by adding $2x + 3 - 2y + 2$.

- Note that the forward differences of a 1$^{\text{st}}$ degree polynomial were constants and those of a 2$^{\text{nd}}$ degree polynomial are 1$^{\text{st}}$ degree polynomials; this “first order forward difference,” like a partial derivative, is one degree lower. Let’s make use of this property.
Second Differences (1/2)

- The function $\Delta_E(x, y) = 2x + 3$ is linear, and hence amenable to incremental computation, viz:

  $$\Delta_E(x + 1, y) - \Delta_E(x, y) = 2$$

  $$\Delta_E(x + 1, y - 1) - \Delta_E(x, y) = 2$$

- Similarly

  $$\Delta_{SE}(x + 1, y) - \Delta_{SE}(x, y) = 2$$

  $$\Delta_{SE}(x + 1, y - 1) - \Delta_{SE}(x, y) = 4$$
Second Differences (2/2)

• So for any step, we can compute new \( \Delta_E(x,y) \) from old \( \Delta_E(x,y) \) by adding an appropriate second constant increment – we update the update terms as we move.

• Having previously drawn pixel \((a,b)\), in current iteration we decide between drawing pixel at \((a + 1,b)\) and \((a + 1, b - 1)\), using previously computed \(d(a,b)\).

• Having drawn the pixel, we must update \(d(a,b)\) for use next time; we’ll need to find either \(d(a + 1,b)\) or \(d(a + 1,b - 1)\) depending on which pixel we chose.

• Will require adding \( \Delta_E(a,b) \) or \( \Delta_{SE}(a,b) \) to \(d(a,b)\).

• So we…

• Look at \(d(a,b)\) to decide which to draw next, update \(x\) and \(y\).

• Update \(d\) using \( \Delta_E(a,b) \) or \( \Delta_{SE}(a,b) \)

• Update each of \( \Delta_E(a,b) \) and \( \Delta_{SE}(a,b) \) for future use

• Draw pixel
Midpoint Eighth Circle Algorithm

MEC (R) /*1/8th of a circle w/ radius R*/
{
    int x = 0, y = R;
    int delta_E, delta_SE;
    float decision;
    delta_E = 2*x + 3;
    delta_SE = 2*(x-y) + 5;
    decision = (x+1)*(x+1) +
               (y + 0.5)*(y + 0.5) - R*R;
    Pixel(x, y);
    for(; x < y; x++){
        if (decision > 0) { /* Move east */
            decision += delta_E;
            delta_E += 2; delta_SE += 2;
        }
        else { /* move SE */
            y--;
            decision += delta_SE;
            delta_E += 2; delta_SE += 4;
        }
        Pixel(x, y);
    }
}
Analysis

- Uses a float!
- 1 test, 3 or 4 additions per pixel.
- Initialization can be improved
- Multiply everything by 4 ===> No Floats!

Questions

- Are we getting all pixels whose distance from the circle is less than ½?
- Why is “x < y” the right stopping criterion?
- What if it were an ellipse?
Other Scan Conversion Problems

- Patterned primitives
- Aligned Ellipses
- Non-integer primitives
- General conics
Patterned Lines

- Patterned line from P to Q is not same as patterned line from Q to P.

- Patterns can be geometric or cosmetic
  - Cosmetic can be from a background or a pattern sequence.
Aligned Ellipses

- Equation is

\[
\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1
\]

i.e,

\[
b^2x^2 + a^2y^2 = a^2b^2
\]

- Computation of \( \Delta_E \) and \( \Delta_{SE} \) is similar
- Only 4-fold symmetry
- When do we stop stepping horizontally and switch to vertical?
Direction Changing Criterion (1/2)

- When the absolute value of the slope of the ellipse is more than 1, viz:

![Diagram of an ellipse with vectors](image)

- How do you check this? At a point \((x,y)\) for which \(F(x,y) = 0\), a vector perpendicular to the level set is \(\nabla F(x,y)\) which is

\[
\left[ \frac{\partial F}{\partial x}(x, y), \frac{\partial F}{\partial y}(x, y) \right]
\]

- This vector points more **right** than **up** when

\[
\frac{\partial F}{\partial x}(x, y) - \frac{\partial F}{\partial y}(x, y) > 0
\]
Direction Changing Criterion (2/2)

- In our case,
  \[ \frac{\partial F}{\partial x}(x, y) = 2a^2 x \]
  and
  \[ \frac{\partial F}{\partial y}(x, y) = 2b^2 x \]
  so we check for
  \[ 2a^2 x - 2b^2 y > 0 \]
  i.e.
  \[ a^2 x - b^2 y > 0 \]

- This, too, can be computed incrementally.
• Now in ENE octant, not ESE octant. This problem is due to aliasing – much more on this later.
Non – Integer Primitives

• Initialization is harder

• Endpoints are hard, too.
  – Making Line (P,Q) and Line (Q,R) join properly is a good test.

• Symmetry is lost
  – HW: find a non-integer circle which, when scanned by “by-eye” midpoint algorithm, exhibits no symmetry.
General Conics

• Very hard--the octant-changing test is tougher, the difference computations are tougher, etc. Do it only if you have to.

• Note that when we get to drawing gray-scale conics, we will find that this task is easier than drawing B/W conics. If we had solved this problem first, life would have been easier.