RASTERIZING CURVES OTHER THAN LINES

- When dealing with other kinds of curves, we can draw it in one of the following ways:
  - Use explicit form (i.e., \( y = f(x) \)) or a parametric form (i.e., \( y = f(s) \) and \( x = g(s) \)) \( \rightarrow \) approximate curve with line segments
    - Parametric form better for this \( \rightarrow \) easy to get evenly-spaced points along curve path
    - Can connect points OR use best-fitting line (linear regression or least squares approach)
  - Use implicit form (i.e., \( f(x,y) = 0 \)) and derive incremental midpoint method

- In the next few slides, we’ll look at how to rasterize:
  - Circles
  - (Axis-aligned) Ellipses
  - ...using incremental midpoint methods
INTRODUCTION

- A circle can be defined by its **implicit form**:
  \[ f(x, y) = (x - x_c)^2 + (y - y_c)^2 - r^2 = 0 \]
  - \((x_c, y_c)\) → center of circle
  - \(r\) = radius
- Since a circle is symmetric in all 8 octants → just compute one octant and replicate in others
DECISION VARIABLE

• Assume the circle is centered at (0,0)
• We’re going to start by:
  • Incrementing x by 1
  • Choose whether to go down or not in y
• To determine our next choice, we will look at our decision variable based on the midpoint $(x_k + 1, y_k - \frac{1}{2})$:

  • If $p_k < 0$ → midpoint inside circle → choose $y_k$
  • If $p_k > 0$ → midpoint outside circle → choose $y_k - 1$

\[
p_k = f\left(x_k + 1, y_k - \frac{1}{2}\right) = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2\]
UPDATING THE DECISION VARIABLE

• If you work through the math (similar to the way we derived our line midpoint algorithm), we also find two different update conditions

• HOWEVER, because the slope along circle is changing → update values depend on current x and y!

• If $p_k < 0$ → add $2(x + 1) + 1$

• If $p_k > 0$ → add $2(x + 1) + 1 - 2(y - 1)$

• We can make this more efficient if we update $2(x+1)$ and $2(y-1)$ incrementally:

\[
2(x_k + 1) = 2x_k + 2 \\
2(y_k - 1) = 2y_k - 2
\]
INITIAL VALUES

• We will start at \( (0,r) \)

• Initial decision variable value:

\[
p_0 = f\left(1, r - \frac{1}{2}\right) = (1)^2 + \left( r - \frac{1}{2} \right)^2 - r^2 = 1 + \left( r^2 - r + \frac{1}{4} \right) - r^2
\]

\[
= \frac{5}{4} - r
\]

• However, if our radius \( r \) is an integer, we can round \( p_0 \) to \( p_0 = 1 - r \), since all increments are integers
MIDPOINT CIRCLE ALGORITHM SUMMARIZED

1. Input radius $r$ and circle center $(x_c, y_c)$; first point = $(x_0, y_0) = (0, r)$

2. Calculate initial decision variable value:
   \[ p_0 = \frac{5}{4} - r \]

3. At each $x_k$, test $p_k$:
   - If $p_k < 0 \rightarrow$ next point is $(x_{k+1}, y_k)$ and:
     \[ p_{k+1} = p_k + 2x_{k+1} + 1 \]
   - Otherwise $\rightarrow$ next point is $(x_{k+1}, y_k - 1)$ and:
     \[ p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1} \]
   
   Where:
   \[ 2x_{k+1} = 2x_k + 2 \]
   \[ 2y_{k+1} = 2y_k - 2 \]

4. For each calculated position $(x,y)$, plot $(x + x_c, y + y_c)$

5. Plot corresponding symmetric points in other seven octants

6. Repeat steps 3 through 5 UNTIL $x \geq y$

   Effectively pretending we are centered on $(0,0)$
MIDPOINT ELLIPSE ALGORITHM
MIDPOINT ELLIPSE ALGORITHM

• We’re going to assume the following about our ellipse:
  • Axis aligned
  • Centered at (0,0) \(\rightarrow\) won’t affect the pixels we choose

• Ellipses \(\rightarrow\) symmetric in each QUADRANT
  • So, we just need one quadrant’s pixels and then repeat for the other three

• HOWEVER, when the \(\text{abs}(\text{slope}) = 1.0\) \(\rightarrow\) should switch from incrementing in X to decrementing in Y!
  • Means we will have two regions to worry about
ELLIPSE FORMULA AND SLOPE

• We assume the ellipse is centered at (0,0)
• Also, we will multiply our original ellipse formula by \( r_x^2 \) and \( r_y^2 \) to get:

\[
f(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 = 0
\]

• We will also need to check the slope \( \rightarrow \) do implicit differentiation to get \( \frac{dy}{dx} \):

\[
\frac{dy}{dx} = -\frac{2r_y^2 x}{2r_x^2 y}
\]
TESTING THE SLOPE

• When the slope = -1.0 (at boundary of R1 and R2), then:

\[
\frac{dy}{dx} = - \frac{2r_y^2 x}{2r_x^2 y} = -1
\]

\[
2r_y^2 x = 2r_x^2 y
\]

• When the slope < -1.0 (entering R2):

\[
- \frac{2r_y^2 x}{2r_x^2 y} < -1
\]

\[
2r_y^2 x \geq 2r_x^2 y
\]
DEcision Variable: Region 1

- Our decision variable for region 1 is going to be based on where \((x_k + 1, y_k - \frac{1}{2})\) is relative to the “true” ellipse:

\[
p_{1k} = f \left( x_k + 1, y_k - \frac{1}{2} \right)
\]

\[
= r_y^2 (x_k + 1)^2 + r_x^2 \left( y_k - \frac{1}{2} \right)^2 - r_x r_y^2
\]

- Similar to the midpoint circle algorithm:
  - If \(p_{1k} < 0\) \(\rightarrow\) inside ellipse \(\rightarrow\) choose \(y_k\)
  - Otherwise \(\rightarrow\) outside ellipse \(\rightarrow\) choose \(y_k - 1\)

- Initial value:

\[
p_{10} = f \left( 1, r_y - \frac{1}{2} \right) = r_y^2 - r_x r_y + \frac{1}{4} r_x^2
\]
INCREMENTING THE DECISION VARIABLE: REGION 1

• If $p_{1k} < 0 \rightarrow$ chose $y_{k+1} = y_k \rightarrow$

$$p_{1k+1} = p_{1k} + 2r^2_y (x_k + 1) + r^2_y$$

• Otherwise $\rightarrow$ chose $y_{k+1} = y_k - 1 \rightarrow$

$$p_{1k+1} = p_{1k} + 2r^2_y (x_k + 1) + r^2_y - 2r^2_x (y_k - 1)$$
WHEN DO WE SWITCH TO REGION 2?

- We have to check when this expression becomes true: \(2r_y^2 x \geq 2r_x^2 y\)

- At the starting point \((0, r_y)\), these values are:
  \[
  s_x = 2r_y^2 x = 0 \\
  s_y = 2r_x^2 y = 2r_x^2 r_y
  \]

- So, every iteration:
  - Add \(2r_y^2\) to \(s_x\)
  - If we choose \((y – 1)\), subtract \(2r_x^2\) from \(s_y\)
DECISION VARIABLE: REGION 2

\[ p_{2k} = f\left(x_k + \frac{1}{2}, y_k - 1\right) = r_y^2\left(x_k + \frac{1}{2}\right)^2 + r_x^2(y_k - 1)^2 - r_x^2r_y^2 \]

- Decrementing in \( y \)

- If \( p_{2k} > 0 \) → outside → choose \( x_k \) → increment \( p_{2k} \):
  \[ p_{2k+1} = p_{2k} - 2r_x^2(y_k - 1) + r_x^2 \]

- Otherwise → inside → choose \( (x_k + 1) \) → increment \( p_{2k} \):
  \[ p_{2k+1} = p_{2k} - 2r_x^2(y_k - 1) + r_x^2 + 2(x_k + 1)r_y^2 \]
INITIAL DECISION VARIABLE: REGION 2

• We’ll use the last point we had before switching to region 2:

\[
p^{20} = f\left(x_0 + \frac{1}{2}, y_0 - 1\right)
\]

\[
= r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2
\]

• An alternative would be to start at \((r_x, 0)\) and go counterclockwise; we would then INCREMENT \(y\) and choose between \(x\) and \((x - 1)\)
MIDPOINT ELLIPSE ALGORITHM: REGION 1

1. Input $r_x$, $r_y$, and ellipse center $(x_c, y_c)$; get first point for ellipse centered on origin: $(x_0, y_0) = (0, r_y)$

2. Get initial decision variable for region 1:

$$p_{10} = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3. While $(2r_y^2 x < 2r_x^2 y)$
   - If $p_{1k} < 0 \Rightarrow$ choose $(x_k + 1, y_k)$
   - Otherwise $\Rightarrow$ choose $(x_k + 1, y_k - 1)$

$$p_{1k+1} = p_{1k} + 2r_y^2 (x_k + 1) + r_y^2$$

$$p_{1k+1} = p_{1k} + 2r_y^2 (x_k + 1) + r_y^2 - 2r_x^2 (y_k - 1)$$
MIDPOINT ELLIPSE ALGORITHM: REGION 2

4. Get initial value of decision variable for region 2 from last position \((x_0, y_0)\) calculated in region 1:

\[ p_{20} = r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2(y_0 - 1)^2 - r_x^2r_y^2 \]

5. While \((y \neq 0)\)
   - If \(p_{2k} > 0\) \(\rightarrow\) choose \((x_k, y_k - 1)\)
   - Otherwise \(\rightarrow\) choose \((x_k + 1, y_k - 1)\)

6. For both regions, define symmetry points: \((x, y), (-x, y), (x, -y),\) and \((-x, -y)\)

7. For each position \((x, y)\), plot:

\[
\begin{align*}
x &= x + x_c \\
y &= y + y_c
\end{align*}
\]
We have two ways to deal with non-standard ellipses (i.e., ellipses that do not line up with the coordinate axes):

1) Adapt formulas to work over ENTIRE ellipse path
2) Get the pixels for the standard ellipse, but then transform the points (i.e., rotation matrix)
RASTERIZING CONIC SECTIONS SUMMARY
DRAWING CONIC SECTIONS SUMMARY

• Figure out where the curve is symmetrical
  • Circles $\rightarrow$ symmetrical across 8 octants
  • Ellipses/Hyperbolas $\rightarrow$ symmetrical across 4 quadrants
  • Parabolas $\rightarrow$ symmetrical across vertical axis
• Compute decision variable formulas for areas where slope is less than 1.0 and greater than 1.0
• Go through midpoint algorithm, checking when to switch because of slope