**Outline**

- Scan conversion of circles
- Clipping circles
- Scan conversion of ellipses

---

**Scan Conversion of Circles**

- Generalization of the line algorithm
- Assumptions:
  - circle at (0,0)
  - Fill 1/8 of the circle, then use symmetry

---

**Scan Conversion of Circles**

- Implicit representation of the circle function:
  \[ F(x,y) = x^2 + y^2 - R^2 = 0. \]
  - Note: \( F(x,y) < 0 \) for points inside the circle, and \( F(x,y) > 0 \) for points outside the circle
- Assume we finished pixel \( y, y_1 \)
- What pixel to draw next? (going clockwise)
  - Note: the slope of the circular arc is between 0 and -1
    - Hence, choice is between \( E \) and \( SE \)
    - Idea: If the circle passes above the midpoint \( M \), then we go to \( E \) next, otherwise we go to \( SE \)

---

**Scan Conversion of Circles**

- We need a decision variable \( D \):
  \[ D = F(M) = (y + 1, y_1 - \frac{1}{2}) \]
  \[ = (y + 1)^2 + (y_1 - \frac{1}{2})^2 - R^2 \]
  - If \( D < 0 \) then \( M \) is below the arc, hence the \( E \) pixel is closer to the line.
  - If \( D \geq 0 \) then \( M \) is above the arc, hence the \( SE \) pixel is closer to the line.
Case I: When $E$ is next
- What increment for computing a new $D$?
- Next midpoint is: $(x_p + 2, y_p - (1/2))$
  \[ D_{\text{new}} = f(x_p + 2, y_p - 1 - (1/2)) \]
  \[ = (x_p + 2)^2 + (y_p - 1/2)^2 - R^2 \]
  \[ = (x_p + 2y_p + 4) + (y_p - 1/2)^2 - R^2 \]
  \[ = (x_p + 2y_p + 4) + (y_p - 1/2)^2 - R^2 \]
  \[ = D + (2x_p + 3). \]
- Hence, increment by: $(2x_p + 3)$

Case II: When SE is next
- What increment for computing a new $D$?
- Next midpoint is: $(x_p + 2, y_p - 1 - (1/2))$
  \[ D_{\text{new}} = f(x_p + 2, y_p - 1 - (1/2)) \]
  \[ = (x_p + 2)^2 + (y_p - 1/2)^2 - R^2 \]
  \[ = (x_p + 2y_p + 4) + (y_p - 1/2)^2 - R^2 \]
  \[ = (x_p + 2y_p + 4) + (y_p - 1/2)^2 - R^2 \]
  \[ = D + (2x_p + 3). \]
- Hence, increment by: $(2x_p + 3)$

Scan Conversion of Circles
- How to compute the initial value of $D$:
  - We start with $x = 0$ and $y = R$, so the first midpoint is at $x = 1, y = R-1/2$.
  \[ D_{\text{init}} = f(1, R-1/2) \]
  \[ = 1 + (R-1/2)^2 - R^2 \]
  \[ = 1 + R^2 - R - R^2 \]
  \[ = 5/4 - R. \]

Circle Scan Conversion Algorithm
- Given radius $R$ and center $(0, 0)$
  - First point $\rightarrow (0, R)$
  - Initial decision parameter $D = 1 - R$
  - While $x \leq y$
    - If ($D < 0$)
      - $x++; D += 2x + 3$;
    - else
      - $x++; y--; D += 2(x - y) + 5$
  - WritePoints(x,y)

WritePoints(x,y)
- Writes pixels to the seven other octants
Clipping Circles

- Accept/Reject test: Does bounding box of the circle intersect with clipping box?
- If yes, condition pixel write on clipping box inside/outside test.

Properties of Ellipses

- "Elongated circle": For all points on ellipse, sum of distances to foci is constant
  \[ d_1 + d_2 = \text{const} \]
- If \( F_1 = (x_1, y_1) \) and \( F_2 = (x_2, y_2) \) then
  \[
  \sqrt{(x-x_1)^2 + (y-y_1)^2} + \sqrt{(x-x_2)^2 + (y-y_2)^2} = \text{const}
  \]

Properties of Ellipses

- Equation simplified if ellipse axis parallel to coordinate axis
- Parametric form
  \[
  x = x_c + r_x \cos \theta \\
  y = y_c + r_y \sin \theta
  \]

Symmetry Considerations

- 4-way symmetry
- Unit steps in x until reach region boundary
- Switch to unit steps in y
- Step in x while \( r_x^2 x < r_y^2 y \)
- Switch to steps in y when \( r_y^2 y \geq r_x^2 x \)

Midpoint Algorithm (initializing)

- Similar to circles
  - The initial value for region 1
    \[
    D_{\text{init}} = f(x_p, y_p) = r_y^2 y_p + \frac{1}{4}  \]
  - The initial value for region 2
    \[
    D_{\text{init}} = f(x_p + 1, y_p - 1) = r_y^2 (y_p - 1)^2 + \frac{1}{4}  \]
- Computing the decision variable
  \[
  D = f(x_p + 1, y_p - 1) = r_y^2 (x_p + 1)^2 + r_x^2 (y_p - 1)^2 - r_x^2 r_y^2
  \]
- If \( D < 0 \) then M is below the arc, hence the E pixel is closer to the line.
- If \( D \geq 0 \) then M is above the arc, hence the SE pixel is closer to the line.

Making a Decision
Midpoint Algorithm for Ellipses

### Region 1
- Set first point to (0, r)
- Set the Decision variable to
  \[ D_{new} = r'_x + 2r'_y = r'_x + 2r'_y \]
- Loop (\( n = 1 \))
  - If \( D < 0 \) then pick E and
    \[ D' = r'_x + r'(2y, 3) \]
  - If \( D = 0 \) then pick SE and
    \[ D' = r'_x + r'(-2y, 3) \]
- Until \( 2y < 2 \)

### Region 2
- Set first point to the last computed
- Set the Decision variable to
  \[ D_{new} = r'_x + 2r'_y = r'_x + 2r'_y \]
- Loop (\( n = 1 \))
  - If \( D < 0 \) then pick SE and
    \[ D' = r'_x + r'(2y, 3) \]
  - If \( D = 0 \) then pick S and
    \[ D' = r'_x + r'(-2y, 3) \]
- Until \( y < 0 \)
- Use symmetry to complete the ellipse

### Computing the Increment
- **E case**
  \[ D_{new} = f(y_x + 2y, 3) = r'_x + 2r'_y + 2r'^3 - r'^3 \]
- **SE case**
  \[ D_{new} = f(y_x + 2y, 3) = r'_x + r'(2y, 3) \]
- **Increment**
  \[ \begin{align*}
  &D_{new} < 0 \text{ Increment SE left} \\
  &D_{new} = 0 \quad \text{Done for last pixel} \\
  &D_{new} > 0 \text{ Increment for first pixel}
  \end{align*} \]

### Computing the Increment in 2nd Region
- **Decision variable in 2nd region**
  \[ \begin{align*}
  D &= f(y_x + 2y, 3) = r'_x + 2r'_y + 2r'^3 - r'^3 \\
  &D_{new} = r'_x + r'(2y, 3) + r'(-2y, 3) \\
  &D_{new} = r'_x + r'(2y, 3) + r'(-2y, 3) \\
  &D_{new} = r'_x + r'(2y, 3) + r'(-2y, 3)
  \end{align*} \]
- If \( D < 0 \) then M is left of the arc, hence the SE pixel is closer to the line.
- If \( D > 0 \) then M is right of the arc, hence the S pixel is closer to the line.