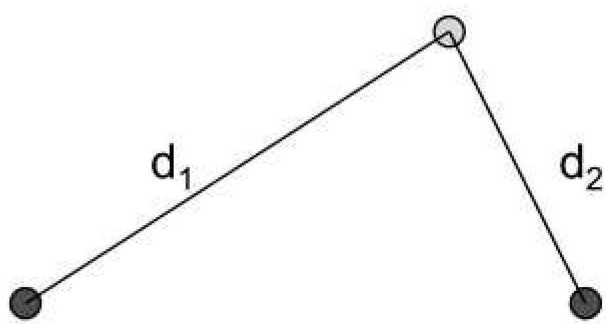
Contents

- Second order curves: ellipse
  - midpoint algorithm
- Geometric proportions
  - Maintaining length
  - Area preservation

Midpoint Algorithm: Ellipse

- Ellipse: collection of points for which the sum of the distances to two given foci is constant: $d_1 + d_2 = \text{const.}$
- Normalized coordinates

$$(x/r_x)^2 + (y/r_y)^2 = 1$$

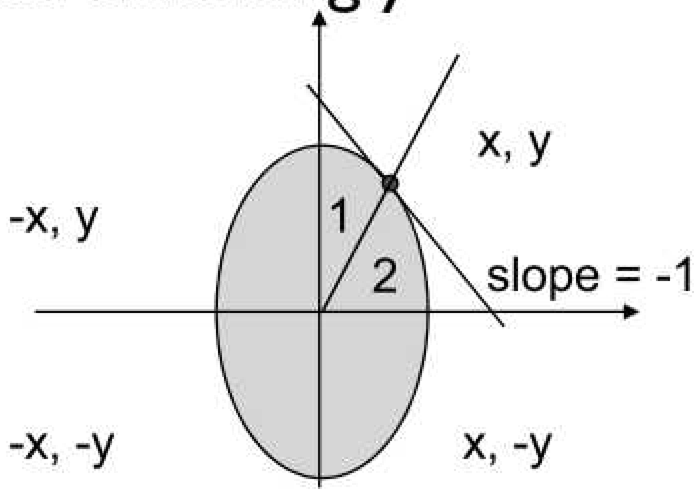


Midpoint Algorithm: Ellipse

- General ellipse: translated and rotated normalized ellipse
- Equation for general ellipse (circle):

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0,\ C^2 - 4AB < 0$$

- Draw one quadrant and use symmetry!
- Start at x = 0, increment x until dy/dx = -1, then switch to decrementing y



Midpoint Algorithm: Ellipse

- Condition for $dy/dx = -1$: occurs at (x_0,y_0) where $x_0*(r_y)^2 = y_0*(r_x)^2$
- At this point we switch from incrementing x to decrementing y!
- Algorithm:

  while ($x*(r_y)^2 < y*(r_x)^2$) $\Delta x = 1$;

  while ($y > 0$) $\Delta y = -1$;

Midpoint Algorithm: Ellipse

- $F_E(x,y) = (r_y)^2 * x^2 + (r_x)^2 * y^2 - (r_x)^2 * (r_y)^2$
  - inside the ellipse $F_E(x,y) < 0$
  - on the boundary $F_E(x,y) = 0$
  - outside the ellipse $F_E(x,y) > 0$
- We apply the midpoint algorithm in two regions: Before and after $dy/dx = -1$
  - Region 1: $\Delta x = 1$
  - Region 2: $\Delta y = -1$

Midpoint Algorithm: Ellipse

- Start with Region 1, then switch to Region 2.
- For Region 1, draw the very first pixel at $(0,r_y)$
- Suppose (x_k,y_k) has just been drawn
- Decision parameter for (x_{k+1},y_{k+1}) in Region 1:

$p1_k = F_E(x_k+1,y_k - 0.5)$

$= (r_y)^2*(x_k+1)^2 + (r_x)^2*(y_k-0.5)^2-(r_x)^2*(r_y)^2$

Midpoint Algorithm: Ellipse

- if $p1_k < 0$ then the midpoint for the next x is inside the ellipse implying that y_k is closer to the boundary than y_k-1 $\Rightarrow$ choose $y_{k+1} = y_k$
- if $p1_k > 0$ then the midpoint for the next x is outside the ellipse implying that $y_k -1$ is closer to the boundary than y_k $\Rightarrow$ choose $y_{k+1} = y_k -1$

Midpoint Algorithm: Ellipse

- A neat trick: calculate by how much p_{k+1} will change from p_k! Answer:

$$p1_{k+1} = \begin{cases} p1_k + (r_y)^2 + 2(r_y)^2 x_{k+1} & , p1_k < 0 \\ p1_k + (r_y)^2 + 2(r_y)^2 x_{k+1} - 2(r_x)^2 y_{k+1} & , p1_k \geq 0 \end{cases}$$

Midpoint Algorithm: Ellipse

- In region 2, $\Delta y = -1$ and we check for the midpoint in the x-direction:

$$p2_k = F_E(x_k+0.5, y_k - 1)$$
$$= (r_y)^2*(x_k+0.5)^2 + (r_x)^2*(y_k-1)^2-(r_x)^2*(r_y)^2$$

- if $p2_k > 0$ then the midpoint for the next y is outside the ellipse implying that x_k is closer to the boundary than $x_k+1 \Rightarrow$ choose $x_{k+1} = x_k$
- if $p2_k < 0$ then the midpoint for the next y is inside the ellipse implying that $x_k +1$ is closer to the boundary than $x_k \Rightarrow$ choose $x_{k+1} = x_k +1$

Midpoint Algorithm: Ellipse

- Again calculate $p2_{k+1}$ from $p2_k$:

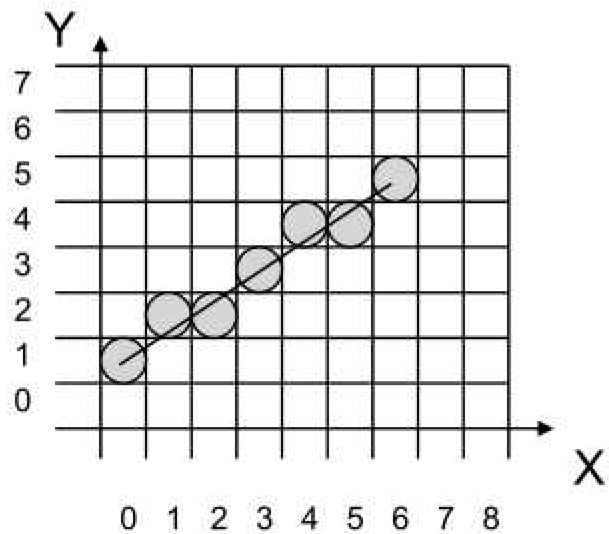
$$p2_{k+1} = F_E(x_{k+1}+0.5, y_{k+1}-1)$$

$$= p2_k - 2(r_x)^2*(y_k-1) + (r_x)^2 + (r_y)^2*[(x_{k+1}+0.5)^2 - (x_k+0.5)^2]$$

- Initial value: last accepted point (x_0, y_0) in Region 1
- Alternatively. start from $(r_x, 0)$ and increment y!!

Maintaining Line Length

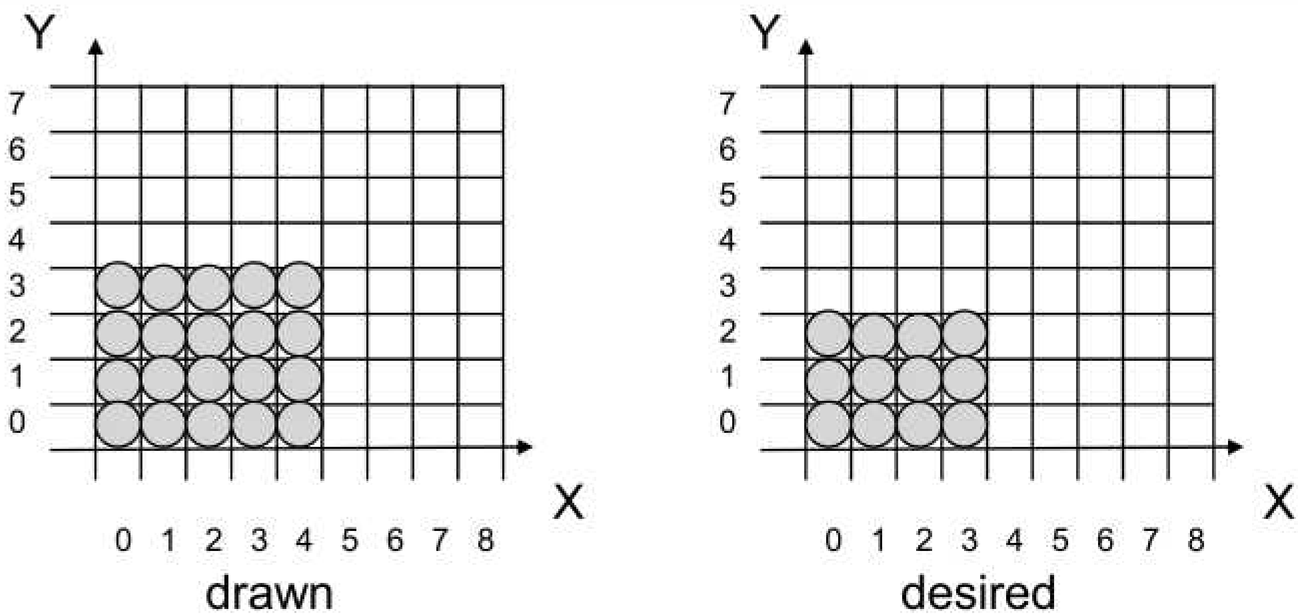
- A straight line plotted with the Bresenham algorithm will yield a line one pixel longer than the original line: (0,1) –> (6,5)
- Possible solution: leave out either end point!



Area preservation

- For a rectangle formed by drawing its perimeter the area will be much too big:

(0,0) –> (4,0) –> (4,3) –> (0,3) –> (0,0)



Area preservation

- Possible solution: a rectangle has an inside and an outside, defined by its mathematical perimeter
- Require pixels to be inside the perimeter!
- This applies to (almost) arbitrary polygons with perimeters of piecewise straight lines as long as we are able to decide whether a given pixel is inside or outside the polygon –> scan line algorithms

Area Preservation: Circle

- The midpoint algorithm looks at pixels which are along the perimeter, not the best pixels inside the circle (see book p. 116).
- Solution: Draw another octant (e.g. the lower left hand quadrant) and use symmetry!