EC-301 Computer Graphics

Lecture Slides- Wk11:

- Setting Frame-Buffer Values
- Circle-Generating Algorithms
- Midpoint Circle Algorithm

Setting Frame-Buffer Values

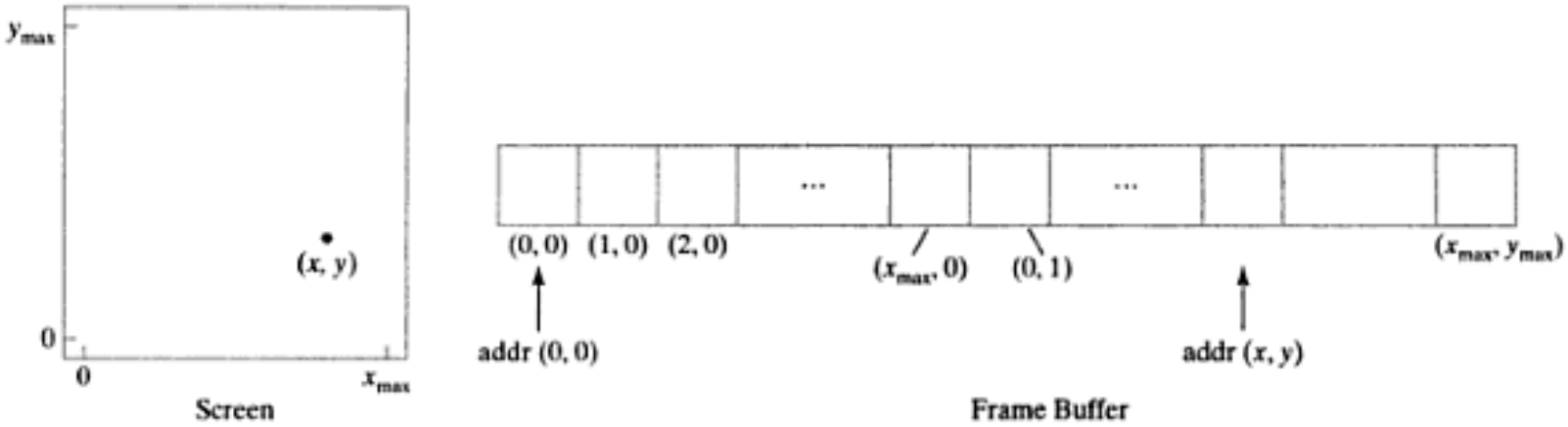


FIGURE 3-14 Pixel screen positions stored linearly in row-major order within the frame buffer.

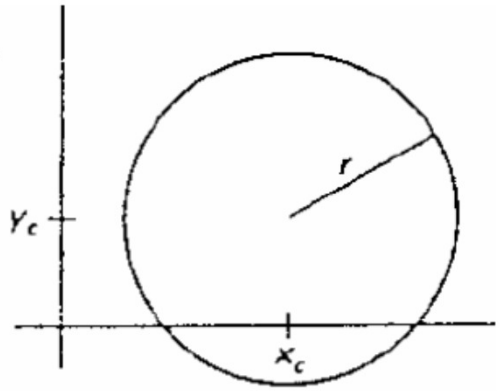
$$addr(x, y) = addr(0, 0) + y.(x_{max} + 1) + x$$

$$addr(x+1, y) = addr(x, y) + 1$$

$$addr(x+1, y+1) = addr(x, y) + x_{max} + 2$$

Circle-Generating Algorithms

- A circle is defined as the set of points that are all at a given distance r from a center point (x_c, y_c)



- For any circle point (x, y), this distance is expressed by the Equation

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

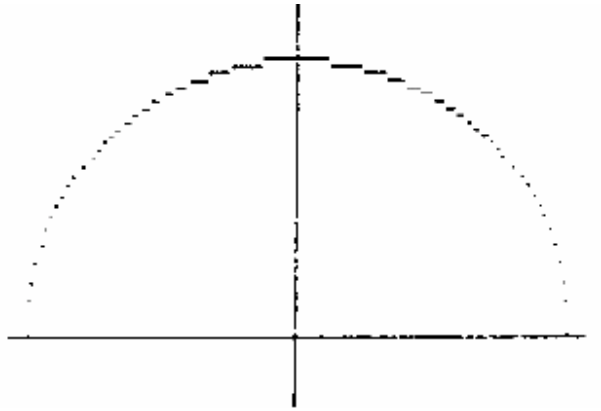
- We calculate the points by stepping along the x-axis in unit steps from x_c-r to x_c+r and calculate y values as

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$

Circle-Generating Algorithms

- **There are some problems with this approach:**

1. Considerable computation at each step.
2. Non-uniform spacing between plotted pixels as in this Figure.



Circle-Generating Algorithms

- Problem 2 can be removed using the polar form:

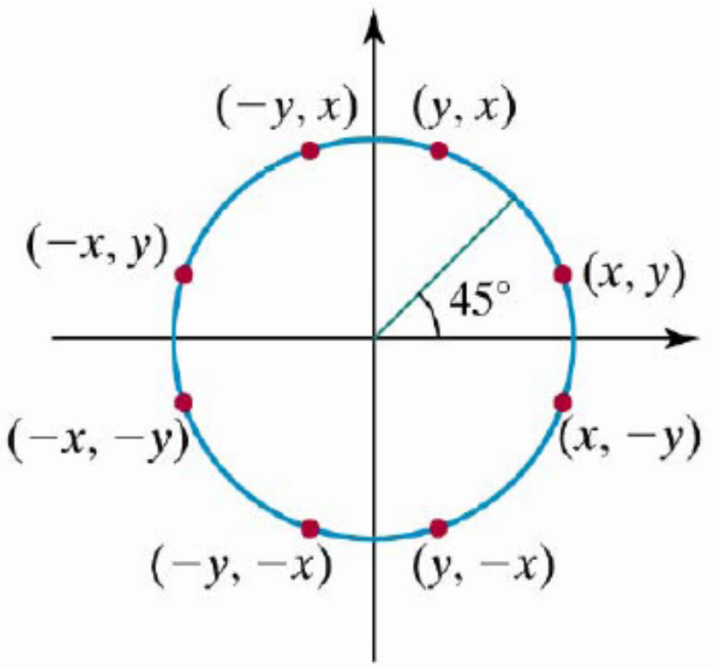
$$x = x_c + r \cos \theta$$

$$y = y_c + r \sin \theta$$

- using a fixed angular step size, a circle is plotted with equally spaced points along the circumference.

Circle-Generating Algorithms

- Problem 1 can be overcome by considering the symmetry of circles as in figure.
- But it still requires a good deal of computation time.



- **Efficient Solutions**
  - Midpoint Circle Algorithm

Midpoint Circle Algorithm

- To apply the midpoint method, we define a circle function:

$$f_{circle}(x,y) = x^2 + y^2 - r^2 = 0 \qquad (2)$$

- Any point (x,y) on the boundary of the circle with radius r satisfies the equation $f_{circle}(x, y)= 0$.

Midpoint Circle Algorithm (Cont.)

- If the point is in the interior of the circle, the circle function is negative.
- If the point is outside the circle, the circle function is positive.

- To summarize, the relative position of any point (x,y) can be determined by checking the sign of the circle function:

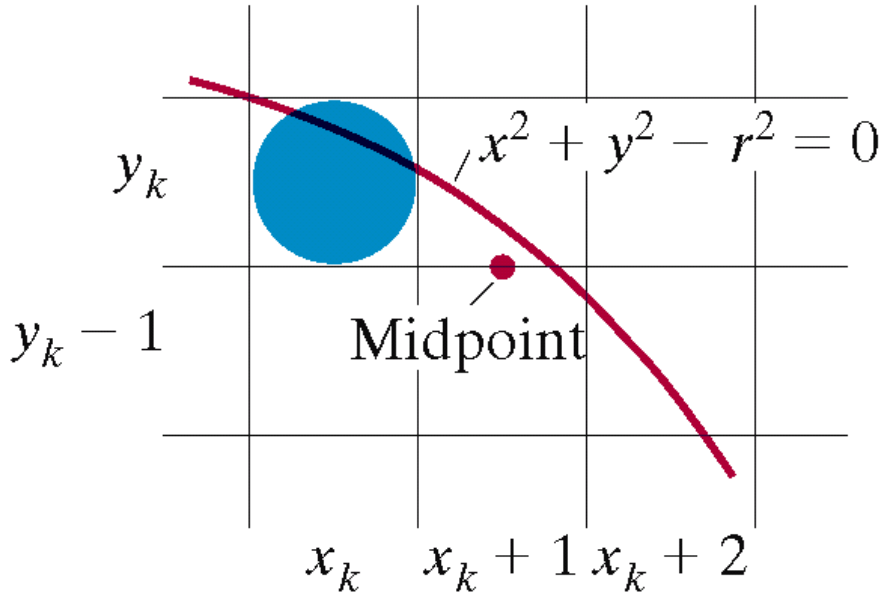
Midpoint Circle Algorithm (Cont.)

$$f_{circle}(x,y) \begin{cases} < 0 & \text{if } (x,y) \text{ is inside the circle boundary} \\ = 0 & \text{if } (x,y) \text{ is on the circle boundary} \\ > 0 & \text{if } (x,y) \text{ is outside the circle boundary} \end{cases} \quad (3)$$

- The circle function tests in (3) are performed for the mid positions between pixels near the circle path at each sampling step. Thus, the circle function is the decision parameter in the midpoint algorithm, and we can set up incremental calculations for this function as we did in the line algorithm.

Midpoint Circle Algorithm (Cont.)

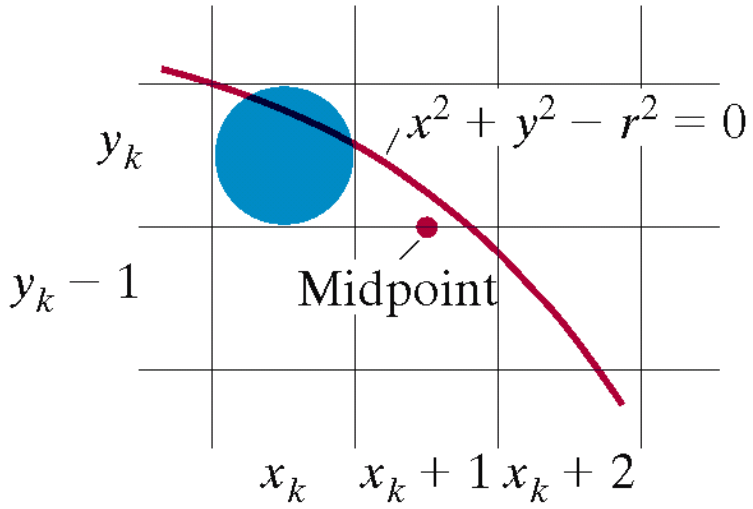
- Figure shows the midpoint between the two candidate pixels at sampling position x_k +1.
- Assuming we have just plotted the pixel at (x_k , y_k), we next need to determine whether the pixel at position (x_k +1, y_k) or the one at position (x_k +1, y_k −1) is closer to the circle.



Midpoint Circle Algorithm (Cont.)

- Our decision parameter is the circle function (2) evaluated at the midpoint between these two pixels:

$$p_k = f_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right)$$
$$= (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \qquad (4)$$



Midpoint Circle Algorithm (Cont.)

- If $p_k < 0$, this midpoint is inside the circle and the pixel on scan line y_k is closer to the circle boundary.
- Otherwise, the midpoint is outside or on the circle boundary, and we select the pixel on scan line $y_k - 1$.
- Successive decision parameters are obtained using incremental calculations.

Midpoint Circle Algorithm (Cont.)

- We obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position $x_{k+1} + 1 = x_k + 2$

$$p_{k+1} = f_{circle}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right)$$

$$= [(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2$$

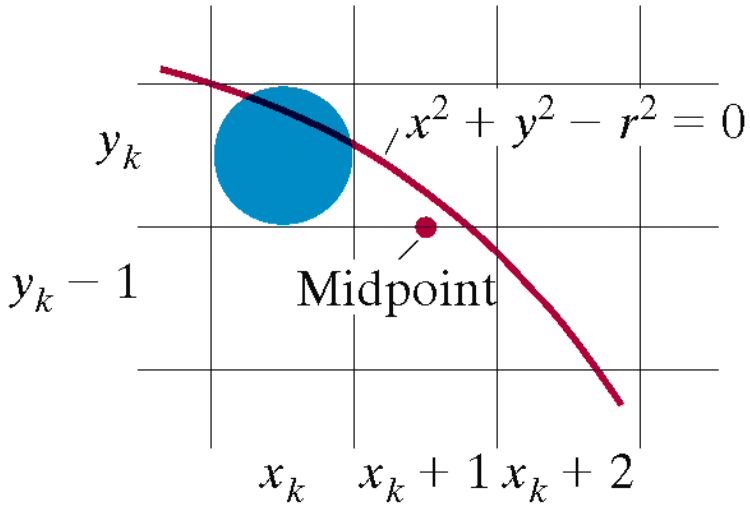
or

$$p_{k+1} = p_k + 2(x_k + 1) + \left(y^2{}_{k+1} - y^2{}_k\right) - (y_{k+1} - y_k) + 1$$

where y_{k+1} is either y_k or y_{k-1},depending on the sign of p_k.

Midpoint Circle Algorithm (Cont.)

- Increments for obtaining p_{k+1} are either
  - $2x_{k+1} + 1$ (if p_k is negative) or
  - $2x_{k+1} + 1 - 2y_{k+1}$ (if p_k is positive)



Midpoint Circle Algorithm (Cont.)

- Increments for obtaining p_{k+1} are either
  - $2x_{k+1} + 1$ (if p_k is negative) or
  - $2x_{k+1} + 1 - 2y_{k+1}$ (if p_k is positive)
- Evaluation of the terms $2x_{k+1}$ and $2y_{k+1}$ can also be done incrementally as:

$$2x_{k+1} = 2x_k + 2$$
$$2y_{k+1} = 2y_k - 2$$

Midpoint Circle Algorithm (Cont.)

- At the start position (0, *r*), these two terms (2x, 2y) have the values 0 and 2*r*, respectively.
- Each successive value is obtained by adding 2 to the previous value of *2x* and subtracting 2 from the previous value of 2*y.*

Midpoint Circle Algorithm (Cont.)

- The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0 , y_0)=(0, r)$:

$$p_0 = f_{circle}\left(1, r-\frac{1}{2}\right)$$

$$= 1+\left(r-\frac{1}{2}\right)^2 - r^2$$

or

$$p_0 = \frac{5}{4} - r$$

Midpoint Circle Algorithm (Cont.)

- If the radius r is specified as an integer, we can simply round p_0 to

$$p_0 = 1 - r \quad (\text{for } r \text{ an integer})$$

- since all increments are integers.

Summary of the Algorithm

- As in Bresenham's line algorithm, the midpoint method calculates pixel positions along the circumference of a circle using integer additions and subtractions, assuming that the circle parameters are specified in screen coordinates. We can summarize the steps in the midpoint circle algorithm as follows.

Algorithm:

1. Input radius r and circle center (x_c, y_c),and obtain the first point on the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$
2. Calculate the initial value of the decision parameter as

$$p_0 = \frac{5}{4} - r$$

3. At each x_k position, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point along the circle centered on $(0,0)$ is (x_{k+1}, y_k) and

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$.

4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_0, y_0) and plot the coordinate values:
   $x = x + x_c, y = y + y_c$

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

Example

- Given a circle radius $r = 10$, we demonstrate the midpoint circle algorithm by determining positions along the circle octant in the first quadrant from $x = 0$ to $x = y$. The initial value of the decision parameter is

$$p_0 = 1 - r = -9$$

Example (Cont.)

- For the circle centered on the coordinate origin, the initial point is (x_0 , y_0) =(0,10), and initial increment terms for calculating the decision parameters are

$$2x_0 = 0, 2y_0 = 20$$

- Successive decision parameter values and positions along the circle path are calculated using the midpoint method as shown in the table (following slide).

Example (Cont.)

k	p_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	$(1,10)$	2	20
1	-6	$(2,10)$	4	20
2	-1	$(3,10)$	6	20
3	6	$(4,9)$	8	18
4	-3	$(5,9)$	10	18
5	8	$(6,8)$	12	16
6	5	$(7,7)$	14	14

If $p_k < 0$, $p_{k+1} = p_k + 2x_{k+1} + 1$

Otherwise, $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$

Example (Cont.)

- A plot of the generated pixel positions in the first quadrant is shown in Figure 5.

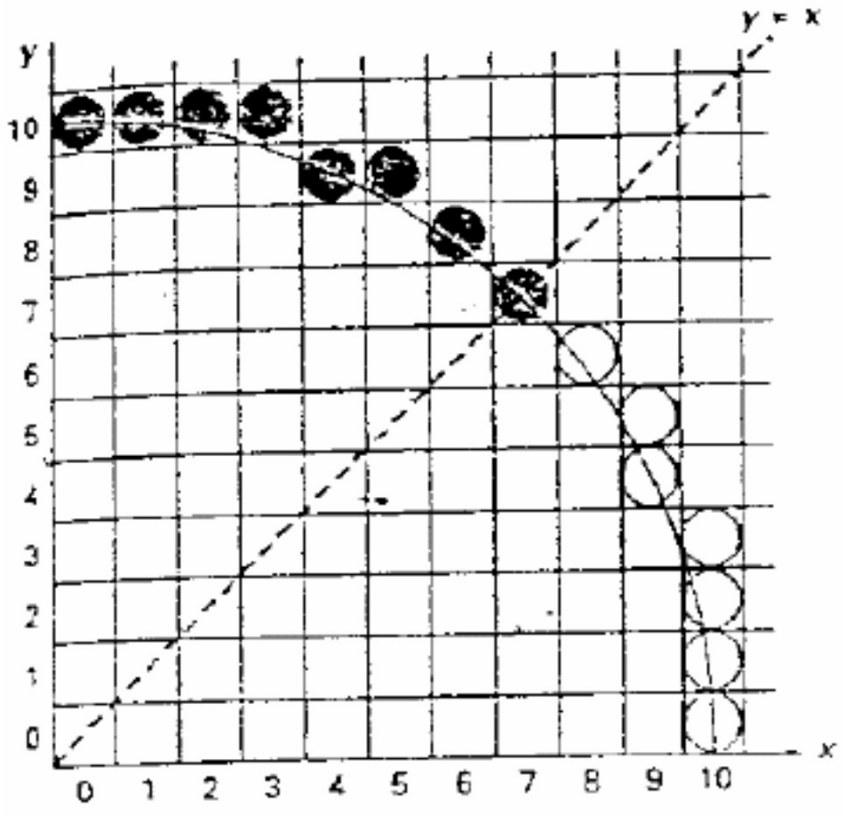


Figure 5