Computer Graphics 2D

3rd class

Lecture 5: Circle Drawing Algorithms

Department of Computer Science

College of Computer and Information Technology

University of Anbar

Assist Prof. Shumoos T. Alfahdawi

1. DDA Circle Drawing Algorithm

The **Digital Differential Analyzer (DDA) Algorithm** is traditionally used for drawing lines in computer graphics. However, it can be adapted to draw circles by using a step-by-step increment approach based on the parametric equations of a circle.

The **DDA Algorithm for drawing circles** calculates points along the circle's circumference by incrementing the angle **θ** in small steps. It uses the parametric form of a circle's equation to compute each point's coordinates.

DDA Circle Algorithm Steps:

1. **Initialize Parameters:**
   - Define the center of the circle as (x_{center}, y_{center})
   - Set the radius r of the circle.
   - Initialize the starting angle $\theta = 0$.
   - Choose a small increment value for θ, such as $d\theta = 1^o$ (or **0.01745** radians).
2. **Calculate Initial Points:**
   - Use the parametric equations of a circle to calculate the initial point on the circumference:

   $$\boxed{\begin{aligned} x &= x_{center} + r \cdot \cos(\theta) \\ y &= y_{center} + r \cdot \sin(\theta) \end{aligned}}$$

   - For **θ=0**, this simplifies to:

   $$\boxed{\begin{aligned} x &= x_{center} + r \\ y &= y_{center} \end{aligned}}$$

   - Plot the initial point (**x, y**) on the screen.
3. **Iterate Over Angles**:
   - Start a loop that will iterate from $\theta = 0$ (or 2π radians) in increments of $d\theta$.
4. **Compute New Points for Each θ Increment**:
   - Each increment of θ, compute the new points using the parametric equations:

   $$\boxed{\begin{aligned} x_{new} &= x_{center} + r \cdot \cos(\theta) \\ y_{new} &= y_{center} + r \cdot \sin(\theta) \end{aligned}}$$

   - Round x_{new} and y_{new} to the nearest integer values to determine the pixel location on the screen.

5. **Plot Points and Use Symmetry:**
    - Plot the calculated point ($\mathbf{x}_{\mathbf{new}}$, $\mathbf{y}_{\mathbf{new}}$) on the screen.
    - Due to the symmetry of a circle, reflect this point across all octants. For example, if you have a point (**x, y**), the symmetric points would be:

$$\mathbf{(x, y),(y, x),(-x, y),(-y, x),(x,-y),(y,-x),(-x,-y),(-y,-x)}$$

    - Plot all these points to complete the circle.

6. **Repeat Until *θ* Covers the Full Circle:**
    - Continue the loop until ***θ*** reaches $\boldsymbol{360^{o}}$ (or **2π** radians).

7. **End of Algorithm:**
    - The loop completes when ***θ*** has covered the entire circle, and the circle is fully drawn on the screen.

Key Points to Note:

- **Angle Increment:** Choosing a smaller increment for θ (e.g., $\mathbf{1^{o}}$ or **less**) will result in a smoother circle but may require more computation.
- **Symmetry:** The use of symmetry helps reduce the number of calculations needed, making the drawing process faster.
- **Rounding:** Since pixel coordinates are integers, rounding is essential to find the nearest pixel position for the computed points.

✓ **Example: Let's draw a circle using DDA Circle Algorithm with circle centered at (0, 0) and a radius of 5.**

1. **Initialize Parameters**:
    - Center of the circle: **(0, 0)**
    - Radius: ***r*=5**
    - Start angle: ***θ*=0°**
    - Angle increment: ***dθ*=15°** (which is approximately **0.2618** radians)

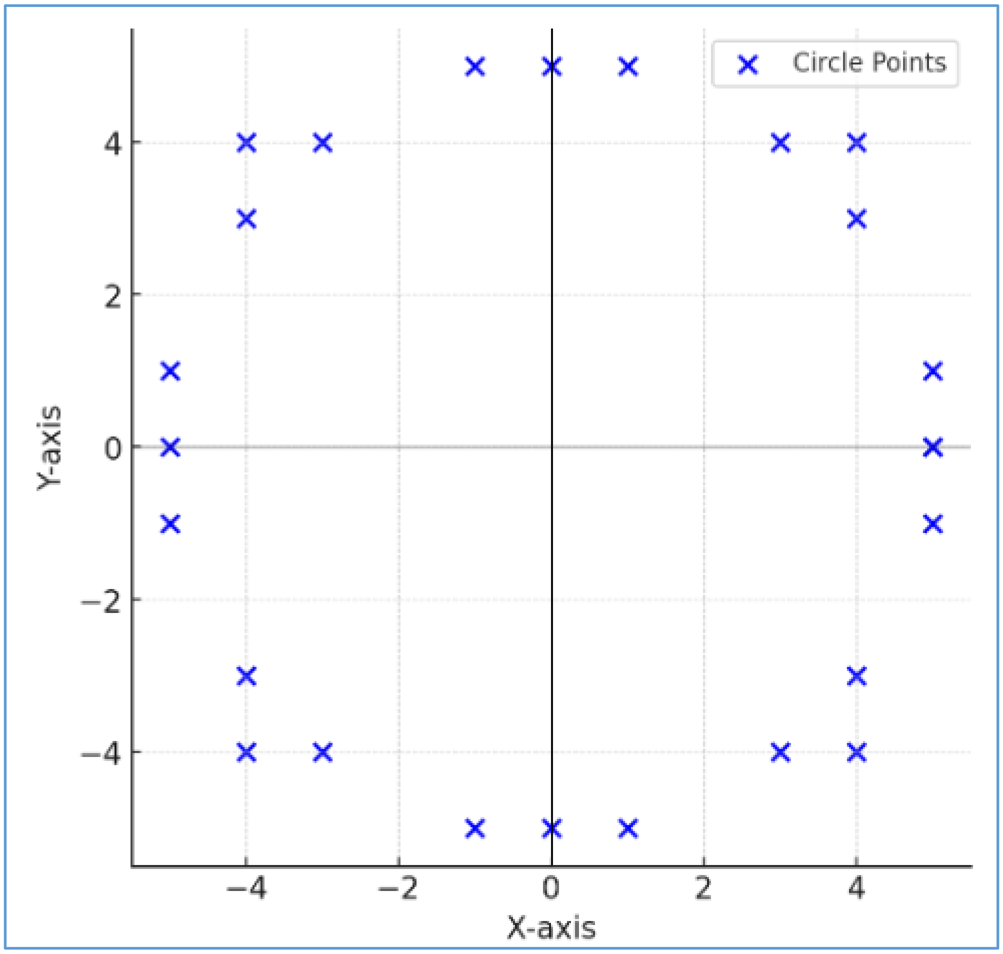
2. **Calculate Points**:
    - Use the parametric equations for the circle:

$$\mathbf{x} = r \cdot \cos(\theta)$$
$$\mathbf{y} = r \cdot \sin(\theta)$$

    - For each increment of ***θ***, compute the new **x** and **y** coordinates, round them to the nearest integer, and plot them.

Table(2): Points Calculated Using the DDA Circle Algorithm

Step	θ (degrees)	θ (radians)	$x = 5 \cdot \cos(\theta)$	$y = 5 \cdot \sin(\theta)$	Rounded (x, y)
1	0	0.0000	5.00	0.00	(5, 0)
2	15	0.2618	4.83	1.29	(5, 1)
3	30	0.5236	4.33	2.50	(4, 3)
4	45	0.7854	3.54	3.54	(4, 4)
5	60	1.0472	2.50	4.33	(3, 4)
6	75	1.3090	1.29	4.83	(1, 5)
7	90	1.5708	0.00	5.00	(0, 5)
8	105	1.8326	-1.29	4.83	(-1, 5)
9	120	2.0944	-2.50	4.33	(-3, 4)
10	135	2.3562	-3.54	3.54	(-4, 4)
11	150	2.6180	-4.33	2.50	(-4, 3)
12	165	2.8798	-4.83	1.29	(-5, 1)
13	180	3.1416	-5.00	0.00	(-5, 0)
14	195	3.4034	-4.83	-1.29	(-5, -1)
15	210	3.6652	-4.33	-2.50	(-4, -3)
16	225	3.9270	-3.54	-3.54	(-4, -4)
17	240	4.1888	-2.50	-4.33	(-3, -4)
18	255	4.4506	-1.29	-4.83	(-1, -5)
19	270	4.7124	0.00	-5.00	(0, -5)
20	285	4.9742	1.29	-4.83	(1, -5)
21	300	5.2360	2.50	-4.33	(3, -4)
22	315	5.4978	3.54	-3.54	(4, -4)
23	330	5.7596	4.33	-2.50	(4, -3)
24	345	6.0214	4.83	-1.29	(5, -1)
25	360	6.2832	5.00	0.00	(5, 0)



Figure(3): visual representation of the points plotted on a coordinate axis.