UNIT-II: Raster Graphics Algorithms and Transformations

1. Raster Graphics Algorithms

Raster graphics algorithms are used to represent and manipulate shapes on raster devices (pixel-based displays). These algorithms work by selecting the appropriate pixels to create geometric shapes like lines, circles, and polygons.

1.1 Line Drawing Algorithms

Importance:
Line drawing algorithms are used to approximate straight lines on raster displays, which are inherently grid-like. These algorithms ensure smooth and accurate rendering.

1.1.1 Digital Differential Analyzer (DDA) Algorithm

- **Definition**:
  The DDA algorithm is an incremental scan-conversion method that calculates intermediate points to draw a line.
- **Key Points**:
  - Uses floating-point arithmetic, making it simple but slower.
  - Steps involve calculating the difference between the endpoints, determining the number of steps, and plotting points incrementally.
- **Advantages**: Easy to implement.
- **Disadvantages**: Inefficient due to floating-point operations.

1.1.2 Bresenham’s Line Algorithm

- **Definition**:
  A faster algorithm that uses only integer calculations to decide which pixel to illuminate.
- **Key Points**:
  - Based on the concept of an error term to determine the closest pixel.
  - Efficient and widely used in computer graphics.
- **Advantages**: Faster and more accurate than DDA.
- **Disadvantages**: More complex to implement.

1.2 Circle and Ellipse Drawing Algorithms

Purpose:
To draw circles and ellipses accurately by using their mathematical properties and exploiting symmetry.

1.2.1 Midpoint Circle Algorithm

- **Definition**:
  Determines the pixels closest to a circle's boundary using a decision parameter.
- **Key Features**:
  - Uses symmetry to reduce calculations.
  - Efficient integer-based calculations.

1.2.2 Ellipse Drawing Algorithm

- **Definition**:
  Extends the midpoint algorithm to handle ellipses using the general ellipse equation.
- **Key Points**:
  - Divides calculations into two regions for efficiency.
  - Mirrors points across axes to complete the ellipse.

1.3 Filling Algorithms

Purpose:
Filling algorithms are used to color the interior of shapes, such as polygons or regions bounded by curves.

1.3.1 Scan-Conversion Polygon Filling

- **Definition**:
  Processes horizontal scan lines to determine which pixels lie inside a polygon.

1.3.2 Inside-Outside Test

- **Definition**:
  Determines whether a point lies inside a polygon by extending a ray and counting edge intersections.

1.3.3 Boundary Fill Algorithm

- **Definition**:
  Recursively fills a region until a boundary color is encountered.

1.3.4 Flood Fill Algorithm

- **Definition**:
  Recursively fills all connected pixels of a specified color.

---

2. Transformations

Transformations are operations that change the position, size, orientation, or shape of graphical objects. They are essential in computer graphics for modeling and rendering.

2.1 2D Transformations

2D transformations manipulate objects within a 2D plane. These include:

- **Translation**: Moves an object by a certain distance.
- **Rotation**: Rotates an object around a point by an angle.
- **Reflection**: Produces a mirror image of an object.
- **Shearing**: Skews the shape in horizontal or vertical directions.
- **Scaling**: Changes the size of an object proportionally or non-proportionally.

Mathematical Representation:
Transformations are represented using mathematical formulas for and coordinates. For example:

- Translation:
- Rotation:

2.2 Homogeneous Coordinate Representation

Definition:
Homogeneous coordinates introduce an extra coordinate to simplify the representation of transformations as matrix operations.

Advantages:

- Allows multiple transformations to be combined into a single matrix.
- Simplifies the implementation of complex transformations.

Example (Translation):

2.3 3D Transformations

3D transformations extend the concepts of 2D transformations into three dimensions, adding the - coordinate.

- **Translation**: Moves an object in , , and directions.
- **Scaling**: Changes the size in all three dimensions.
- **Rotation**: Rotates around one of the three axes (, ,).