3.2D Raster Algorithms

3.1 Scan Converting Lines

- find algorithms that can rapidly generate the **pixels** comprising the line in a very accurate way

**Midpoint Line Algorithm**

\[(x_{i-1}, y_{i-1}): \text{pixel plotted when } x = x_{i-1}\]

\[T_i, S_i: \text{only possible pixels to be chosen from when } x = x_i\]
**Question:** which pixel, $T_i$ or $S_i$, should be selected?

**Answer:**

\[
egin{cases} 
T_i & \text{if } M \text{ is below } Q \\
S_i & \text{if } M \text{ is above } Q 
\end{cases}
\]

**How to make efficient decision?**

Implicit line equation: $F(x,y) \equiv ax + y + c = 0$

\[
a = \Delta y \quad (\Delta y \equiv y_{\text{end}} - y_{\text{start}})
\]

\[
b = -\Delta x \quad (\Delta x \equiv x_{\text{end}} - x_{\text{start}})
\]

\[
c = B \cdot \Delta x \quad (B \text{ : y intercept})
\]

Consider the sign of

\[
e_i \equiv F(M) = F(x_i, y_{i-1} + \frac{1}{2})
\]

Then

\[
\begin{cases} 
T_i \text{ is plotted if } e_i \geq 0 \\
S_i \text{ is plotted if } e_i < 0
\end{cases}
\]
or, consider the sign of

\[ E_i \equiv 2F(M) = 2F(x_i, y_{i-1} + \frac{1}{2}) \]

\[ = 2 \cdot \Delta y \cdot x_i - \Delta \cdot (2y_{i-1} + 1) + 2B \cdot \Delta x \]

To compute \( E_i \), note that

\[ E_{i+1} = 2F(x_{i+1}, y_i + \frac{1}{2}) \]

\[ = 2 \cdot \Delta y \cdot x_{i+1} - \Delta \cdot (2y_i + 1) + 2B \cdot \Delta x \]

Therefore

\[ E_{i+1} - E_i = 2\Delta y - 2\Delta x (y_i - y_{i-1}) \]
Since
\[
y_i = \begin{cases} 
y_{i-1} + 1, & \text{if } E_i \geq 0 \\
y_{i-1}, & \text{if } E_i < 0
\end{cases}
\]

we have the following recurrence formula
\[
E_{i+1} = \begin{cases} 
E_i + 2(\Delta y - \Delta x), & E_i \geq 0 \\
E_i + 2\Delta y, & E_i < 0
\end{cases}
\]

with the initial condition
\[
E_1 = 2\Delta y - \Delta x
\]
Example:

Which pixel should be plotted for the line $x = 2$, $x = 6$?
3.2 Scan Converting Circles

Midpoint Circle Algorithm

Consider the portion from $x = 0$ to $x = R/\sqrt{2}$ only
(same assumption as the previous algorithm)

$(x_{i-1}, y_{i-1})$: pixel chosen for $x = x_{i-1}$
$A_i, B_i$: only possible choices for $x = x_i$

**Question:** which pixel should be selected?
Consider the sign of

\[ d_i \equiv D(M) = D\left(\left( x_i, y_{i-1} - \frac{1}{2}\right)\right) \]

where \( D((x, y)) \equiv (x^2 + y^2) - R^2 \)

Then,

\[
\begin{cases} 
B_i \text{ is plotted if } d_i = D(M) \geq 0 \\
A_i \text{ is plotted if } d_i = D(M) < 0
\end{cases}
\]

How to compute \( d_i \) efficiently?

\begin{align*}
    d_i &= x_i^2 + \left( y_{i-1} - \frac{1}{2} \right)^2 - R^2 \\
    d_{i+1} &= x_{i+1}^2 + (y_i - \frac{1}{2})^2 - R^2 \\
    d_{i+1} - d_i &= 2x_{i-1} + 3 + (y_i - y_{i-1})(y_i + y_{i-1} - 1)
\end{align*}

Since

\[
    y_i = \begin{cases} 
        y_{i-1}, & \text{if } d_i < 0 \\
        y_{i-1} - 1, & \text{if } d_i \geq 0
    \end{cases}
\]
it follows that

\[ d_{i+1} = \begin{cases} 
  d_i + 2x_{i-1} + 3, & d_i < 0 \\
  d_i + 2(x_{i-1} - y_{i-1}) + 5, & d_i \geq 0 
\end{cases} \]

with

\[ d_1 = \frac{5}{4} - R \]

We will use the program transformation to eliminate fractions in the above recurrence formula.

Let \( h_i \equiv d_i - 1/4 \) (hence, \( h_1 = d_1 - 1/4 = 1 - R \))

then we have

\[ \begin{cases} 
  d_i \geq 0 \text{ iff } h_i \geq -\frac{1}{4} \\
  d_i < 0 \text{ iff } h_i < -\frac{1}{4} 
\end{cases} \]
Since \( h_{i+1} = d_{i+1} - \frac{1}{4} \), it follows that

\[
h_{i+1} - h_i = d_{i+1} - d_i
\]

and, consequently,

\[
h_{i+1} = \begin{cases} 
  h_i + 2x_{i-1} + 3, & h_i < -\frac{1}{4} \\ 
  h_i + 2(x_{i-1} - y_{i-1}) + 5, & h_i \geq -\frac{1}{4}
\end{cases}
\]

But this is equivalent to

\[
h_{i+1} = \begin{cases} 
  h_i + 2x_{i-1} + 3, & h_i < 0 \\ 
  h_i + 2(x_{i-1} - y_{i-1}) + 5, & h_i \geq 0
\end{cases}
\]

with \( h_1 = 1 - R \).
3.3 Scan Converting Polygons

**Basic idea:**

1. **Compute** \( x \) coordinates of intersection points of current scan line with all edges
2. **Sort** intersection points by increasing \( x \) values
3. **Group** intersection points by pairs
4. **Fill** in the pixels between each pair of intersection points on the current scan line
Need to create a **bucket-sorted edge table (ET)** first:

- To determine which edges intersect current scan line
- To efficiently compute intersection points of these edges with the current scan line
- Vertical edges need to be shortened by 1 in y direction

**Example:**

![Diagram of scan lines and edges](image-url)
Also need to maintain an **active-edge table (AET)**

**Purpose:** keep track of the edges the current scan line intersects

**How:** when we move to a new scan line (bottom to top), new edges intersecting the new scan line are added into the AET, edges in AET which are no longer active (not intersected by the new scan line) are deleted

**Example:**

In the previous example, when $y = 2$,

$$AET \rightarrow$$

when $y = 4$,

$$AET \rightarrow$$

when $y = 8$,

$$AET \rightarrow$$
Algorithms:

1. Set $y$ to the $y$-coordinate of the first non-empty bucket

2. Set AET to empty

3. **Repeat until** the AET and ET are both empty

   3.1 **Merge** edges from ET bucket $y$ with edges in AET, maintaining AET sort order on $x$

   3.2 **Fill in** pixels on scan line $y$ bounded by pairs of $x$-coordinates from edges in AET

   3.3 **Remove** from AET those edges for which $y = y_{top}$

   3.4 For each edge remaining in AET, **replace** $x$ with $x + 1/m$

   3.5 **Increment** $y$ by 1, to the coordinate of the next scan line
3.4 Pattern Filling

- to illuminate only those pixels that correspond to a pattern

**Pattern**: a two dimensional boolean array

Patterns are placed end to end to give the repeating pattern effect

**How to fill a polygon with patterns?**

1. Find (startx, starty): lower-left corner of the polygon
2. For each pixel (currentx, currenty) in the polygon, compute the offset from the starting vertex
3. Determine the corresponding pattern entry for (currentx, currenty)
- 36 -

- How should \((x,y)\) be computed?

- How should the pattern filling process be integrated with the polygon filling process?
3.5 Polygon Filling Revisited

Polygon filling based on vertex classification
(Cheng & Toll)

Basic idea:

- Classify vertices according to the action that needs to be taken during the filling process
- maintain a list of active edges and modify the ordered list of edges based on classification of the vertices encountered
Vertex Classification

**Vertex type:** START, STOP, MERGE, SPLIT, CONTINUE

![Diagram showing vertex types](image)
Sort the vertices (by y) into $v_1, v_2, \ldots, v_n$: 

\[ \begin{array}{c}
\vdash \\
v_8 \\
v_7 \\
v_6 \\
v_4 \\
v_4 \\
v_3 \\
v_2 \\
v_1 
\end{array} \]
The algorithm:

Set AEL to empty;

\( i = 1; \)

While \( ( i \leq n - 1 ) \) do

\[
\text{case ( type of } v_i \text{ ) of}
\]

\( \text{START} : \) add adjacent edges of \( v_i \) to AEL;

\( \text{STOP} : \) remove adjacent edges of \( v_i \) from AEL;

\( \text{MERGE} : \) remove adjacent edges of \( v_i \) from AEL;

\( \text{SPLIT} : \) add adjacent edges of \( v_i \) to AEL;

\( \text{CONTINUE} : \) replace the adjacent edge of \( v_i \) which is in AEL with the other adjacent edge;

\( \text{fill} \) the polygon from \( v_i \) to \( v_{i+1}; \) //see next page

\( i = i+1; \)
**Fill polygon** from $v_i$ to $v_{i+1}$:

Initialize $e.y$ and $e.x$ for each edge $e$ in AEL if not already done;

```plaintext
while (current scan line $y < y$-coordinate of $v_{i+1}$) do
  for each pair of edges $e$ and $e'$ in AEL do
    fill the pixels between $e.x$ and $e'.x$ on the current scan line $y$;
    $e.y \leftarrow e.y + 1$; $e.x \leftarrow e.x + \frac{1}{m}$;
    $e'.y \leftarrow e'.y + 1$; $e'.x \leftarrow e'.x + \frac{1}{m'}$;
    $y \leftarrow y + 1$; //move to the next scan line
```

$m$: slope of edge $e$
$m'$: slope of edge $e'$
$v.x$: $x$-coordinate of $v$
$v.y$: $y$-coordinate of $v$
$e.x$: current $y$ of $e$;
  //initially $e.x$ equals $x$-coordinate of its lower vertex
$e.y$: current $y$ of $e$
  //initially $e.y$ equals $y$-coordinate of its lower vertex