Raster Graphics in X Windows

Professor Raymond Zavodnik

Tbilisi State University April 23, 2013

Drawing a Straight Line

1. Given two integer-valued endpoints (X_A, Y_B) and (X_B, Y_B) calculate the pixel-values that lie nearest to the ideal line
2. The ideal line between these two points has the form $y = mx + b$, where $m = \frac{Y_B - Y_A}{X_B - X_A} = \frac{\Delta y}{\Delta x}$
3. Thus, given successive x-values $x = X_A, \cdots, X_B$ calculate y and round up or down.
4. Problem: This calculation uses floating point calculations

Drawing a Straight Line

1. Given two integer-valued endpoints (X_A, Y_B) and (X_B, Y_B) calculate the pixel-values that lie nearest to the ideal line
2. The ideal line between these two points has the form $y = mx + b$, where $m = \frac{Y_B - Y_A}{X_B - X_A} = \frac{\Delta y}{\Delta x}$
3. Thus, given successive x-values $x = X_A, \cdots, X_B$ calculate y and round up or down.
4. Problem: This calculation uses floating point calculations

Drawing a Straight Line

1. Given two integer-valued endpoints (X_A, Y_B) and (X_B, Y_B) calculate the pixel-values that lie nearest to the ideal line
2. The ideal line between these two points has the form $y = mx + b$, where $m = \frac{Y_B - Y_A}{X_B - X_A} = \frac{\Delta y}{\Delta x}$
3. Thus, given successive x-values $x = X_A, \cdots, X_B$ calculate y and round up or down.
4. Problem: This calculation uses floating point calculations

Drawing a Straight Line

1. Given two integer-valued endpoints (X_A, Y_B) and (X_B, Y_B) calculate the pixel-values that lie nearest to the ideal line
2. The ideal line between these two points has the form $y = mx + b$, where $m = \frac{Y_B - Y_A}{X_B - X_A} = \frac{\Delta y}{\Delta x}$
3. Thus, given successive x-values $x = X_A, \cdots, X_B$ calculate y and round up or down.
4. Problem: This calculation uses floating point calculations

Eliminating Floating Point

- Assume $0 \leq m \leq 1$ and $b = 0$
- Start with $x = 0$
- Problem: Do we pick $(1, 0)$ or $(1, 1)$ as successor?
  - If $\frac{1}{2} \leq \frac{\Delta y}{\Delta x} \leq 1$ pick $(1, 1)$
  - Write this: $e = \frac{\Delta y}{\Delta x} - \frac{1}{2} \geq 0$
  - Similarly: $e = \frac{\Delta y}{\Delta x} - \frac{1}{2} < 0$ means pick $(1, 0)$

- Now consider The next transition. If $(1,0)$ was chosen, *i.e.* $m - 1/2 < 0$, then the next e is $e = 2m - 1/2$. If then $e >= 0$ we must increment y, *i.e.* choose $(2,1)$.
- If this e were less than 0 then y is not incremented. In both cases e is incremented by m, but y is incremented only if $e \geq 0$
- The case $(1,1)$ is handled similarly: If $2 - 2m > 1/2$, or, $0 > m + (m - 1/2) - 1 > 0$ then pick $(2,1)$, thus do not increment y. If $2 - 2m \leq 1/2$ then increment y. In both cases increment e by $m - 1$ and increment y only if e was nonnegative.

Algorithm

1. Assume that $0 \leq m = \frac{\Delta y}{\Delta x} \leq 1$
2. If $e \geq 0$ then increment y. But then decrement e and add m to this: $e \leftarrow e - 1 + m$
3. If $e < 0$ then y is *not* incremented, but $e \leftarrow e + m$
4. Write the initialization of $e = m - 1/2$ as $e' = 2\Delta y - \Delta x$
5. Instead of $e \leftarrow e - 1$ write $e' \leftarrow e' - 2\Delta x$
6. Instead of $e \leftarrow e + m$ write $e' \leftarrow e' + 2\Delta y$

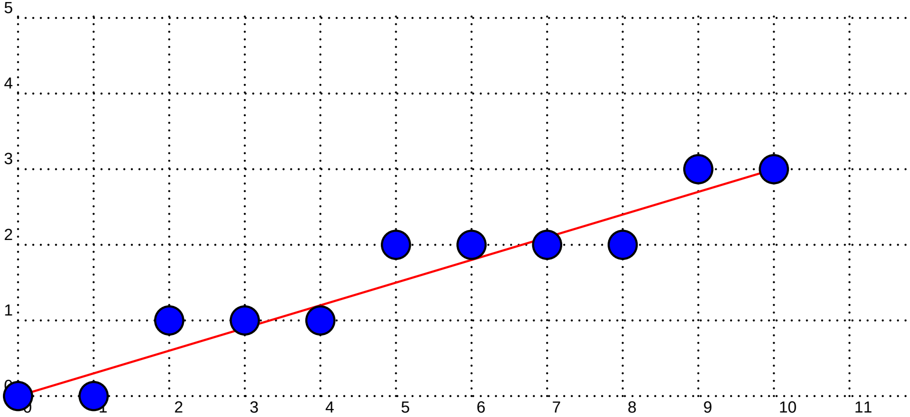
Bresenham's Algorithm

1. $x \leftarrow X_A, y \leftarrow Y_A$
2. $dx \leftarrow X_B - X_A, dy \leftarrow Y_B - Y_A$
3. $e \leftarrow 2dy - dx$
4. for $i = 1$ to dx

    WritePixel(x, y)
    if $e \geq 0$ begin
        $y \leftarrow y + 1$
        $e \leftarrow e - 2dx$
        end
    $x \leftarrow x + 1$
    $e \leftarrow e + 2dy$

end

Example of Line Drawing



$$y = 0.3x$$

Scan Conversion in Raster Graphics

- Rastergraphics enables the filling of closed polygons with pixels of chosen colors
- Almost all modern graphics hardware supports the technology of *Scan Conversion*
- Based on television technology
- Basic algorithm says that when scanning a pixel row from left to right change from background color to polygon color when entering over the first (odd) edge
- Change back to background color when crossing the next (even) edge
- Make sure that pixels do not get set twice at edges
- Thus, use background color when setting boundary pixels on odd edges, set polygon color when exiting over even edges

Integer Arithmetic and Polygon Filling

- Obviously there is also here a need to use integer arithmetic when computing the next pixel of an edge from the previous pixel
- Bresenham does not really work here
- Proceed incrementally, using the coherence of each span of lixels between successive edges to set all pixels in easch span
- Starting with the minimal vertex of a polygon, incrementally calculate using integer arithmetic the intersection of the next scan row with the relevant polygon edges
- Set the pixels to the polygon color on spans between odd intersections

Remarks

- Minimal vertices count twice, inflections once and maximal vertices not at all
- Rule: Approaching an edge from outside from the right round up, from inside from the right round down
- If the left pixel of a span has an integer x-coordinate, it lies inside the polygon, whereas a right such pixel does not
- Use edge coherence to calculate points of intersection between polygon edges and successive scan lines
- Start with: $x_{i+1} = x_i + \frac{1}{m}$, because $\Delta y = 0$ for scan conversion

Elimination of Floating Point

1. Assume $m > 1$ for current edge
2. Write x_i as an integer plus a remainder: $x_i = [x_i] + r_i$ as a sum of an integer $[x_i]$ and a fraction $0 \leq r_i \leq 1$
3. Start with the integer value of some vertex x_{min}
4. Set inc to the numerator of $\frac{1}{m}$
5. Add the numerator to inc until this quantity is greater than the numerator
6. Increment x by 1
7. Reset inc to inc - denom

Algorithm for Polygon Filling

- The **Edgetable** (ET) contains all polygon edges sorted by their smallest y-values. Within a row of the ET sort the edges according to their increasing x-values
- The **Active Edge Table** (AET) is a dynamic data structure that for a given scan line contains those polygon edges that that scan line interesects

Algorithm for Polygon Filling

1. Find minimal vertex (vertices) y_0 of the ET
2. `AET = NULL`
3. if AET `!= NULL` or ET `!=` ϕ
    1. Move edges with $y_{min} = y_0$ into the AET
    2. Sort the AET with increasing x
    3. Write pixel spans for pairs x_1, x_2 in the AET
    4. Remove edges with max coordinates y_0 from the AET
    5. $y \leftarrow y + 1$
    6. Calculate new x-values using integer arithmetic
    7. Resort the AET
4. Go to 3

Polygon Filling Example