Lecture 01: Implementation Algorithms for Primitives (Chapter 6)

I. We will discuss a few algorithms from Chapter 6 in depth for drawing links and polygons and discuss high level implementation of attributes.

II. Line-Drawing

A. Setup

i. Assume integer coordinates for start/endpoint (can acquire by projecting to screen coordinates)

ii. As input we get 2 endpoints of a line \((x_{0}, y_{0}) \rightarrow (x_{end}, y_{end})\)

iii. Recall line equation:

\[ y = mx + b \quad \text{where} \quad m = \frac{y_{end} - y_{0}}{x_{end} - x_{0}} , \quad b = y_{0} - mx_{0} \]

we can compute changes (intervals) from this:

\[ \delta y = m \delta x \quad \text{or} \quad \delta x = \frac{\delta y}{m} \]

iv. To draw line we exploit those properties/equations to determine cells of grid (pixels)
to set in the frame buffer to the line color. Basically, for each \(x\), compute \(y\), until endpoint is reached

B. DDA - Digital Differential Analyzer - Based on computing \(\delta x, \delta y\) for a line

i. For simplicity, assume \(x_{min} \leq x_{0} < x_{end}\)

ii. Let \(\delta x = 1\), then \(y_{k+1} = y_{k} + m\) where \(k\) starts at 0 until endpoint is reached

Then we round \(y_{k}\) to an integer.

iii. For \(m > 1\) we swap roles of \(x\) and \(y\). That is: \(\delta y = 1\) and \(x_{k+1} = x_{k} + \frac{1}{m}\)

iv. Other \(m < 1\) similar equations are derived.

C. Code:

\[ \text{float} \quad x_{inc}, \quad y_{inc}, \quad x_{0}, \quad y_{0} \]

1. int \(dx = x_{end} - x_{0}, \quad dy = y_{end} - y_{0}\), steps, \(K\);
2. float \(x_{inc}, \quad y_{inc}, \quad x_{0}, \quad y_{0}\);
3. if (\(\left| \frac{dx}{|dx|} \right| > |\frac{dy}{dy}|\))
4. steps = \(\left| dx \right|\)
5. else
6. steps = \(\left| dy \right|\)
7. \(x_{inc} = \frac{\delta x}{\text{steps}}, \quad y_{inc} = \frac{\delta y}{\text{steps}}\)
8. \(y = \text{floor}(y_{inc})\) / \text{steps}, \(x = \text{floor}(x_{inc})\) / \text{steps};
9. for (\(x = 0; \quad x < \text{steps}; \quad x++)\)
10. \(x += x_{inc}, \quad y += y_{inc}\);
11. set \((\text{round}(x), \text{round}(y))\).

D. Issues? *Ask class*

- Good: Best for integer to eliminate multiplication.
- Bad: Error in floats accumulates. Float arithmetic is slow.
C. Bresenham's Line Algorithm – accurate fraction (Integer arithmetic only).

i. Observations:
- Only 2 possible y pixels could be chosen, (1, 10) or (1, 11) in example
- Algorithm exploits this with simple test
- Again for simplicity we set $|r| \leq 1$

ii. Idea: Start at $(x_0, y_0)$ go to next x and pick y which is closest to mathematical line; i.e.,

$$x_{k+1} = x_k + 1$$

and we choose $y_{k+1} = y_k$ or $y_{k+1} = y_k + 1$

iii. Derivation,

$$s_0 = m(x_0 + 1) + b$$

$$\text{lower} = y - y_k = m(x_k + 1) + b - y_k$$

$$\text{upper} = y_{k+1} - y = y_k + 1 - m(x_k + 1) - b$$

Proximity is based on sign of lower - upper =

$$= (m(x_k + 1) + b - y_k) - (y_k + 1 - m(x_k + 1) - b)$$

$$= 2m(x_k + 1) - 2y_k + 2b + 1$$

We can substitute $m = \frac{Ay}{\Delta x}$ and multiply by $\Delta x$ (to remove division) to derive the decision $p_k$. It's sign will be the same as lower - upper to define our proximity,

$$p_k = \Delta x (\text{lower} - \text{upper}) = \Delta x (\frac{Ay}{\Delta x} (x_k + 1) - y_k + 1)$$

$$= 2Ay x_k - 2\Delta x y_k + 2\Delta y + \Delta x (\text{lower} - \text{upper}) - \Delta x$$

If $y_k$ is closer, lower < upper than $p_k$ will be negative, otherwise $p_k$ is positive.

So, we can determine $p_{k+1}$ using incremental integer computations,

$$p_{k+1} = 2Ay x_k + 2\Delta y - 2\Delta x y_k + 2\Delta y - \Delta x$$

Note $x_k = x_{k+1} - 1$.

Start $p_0$ = from $p_k$ formula

$$p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x (2(y_0 - \frac{\Delta y}{\Delta x} x_0) - 1)$$

$$= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 3\Delta x y_0 - 2\Delta y - 2\Delta x$$

$$= 2\Delta y - \Delta x$$
IV. Algorithm for $|m| < 1.0$

Input: $x_0, y_0, x_{end}, y_{end}$ as integers

1. set $(x_0, y_0)$

2. set $p = 2dy - dx$

3. $x = x_0, y = y_0, dx = 2y - dx, dy = 2dx - dy$

4. for each $x_k$ along line

5. if $p < 0$

6. set $(x_k, y_k)$

7. $x_{hi} = x_k + 2dy$

8. else

9. set $(x_k, y_k)$

10. $x_{hi} = x_k + 2dx - dy$

a. considers all cases of $m$ through symmetry of octants/quadrents.

b. Advantages: integer arithmetic

V. Example: $(x_0, y_0) = (20, 10), (x_{end}, y_{end}) = (30, 4)$

$\Delta x = 10, \Delta y = 6, p_0 = 2dy - dx = 6, 2dx - dy = 24$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$p_k$</th>
<th>$(x_{hi}, y_{hi})$</th>
<th>$K_j$</th>
<th>$p_j$</th>
<th>$(x_{hi}, y_{hi})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>(21, 11)</td>
<td>5</td>
<td>6</td>
<td>(26, 16)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>(23, 10)</td>
<td>6</td>
<td>3</td>
<td>(29, 16)</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>(25, 10)</td>
<td>7</td>
<td>-3</td>
<td>(29, 16)</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>(24, 13)</td>
<td>8</td>
<td>14</td>
<td>(30, 18)</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>(25, 14)</td>
<td>9</td>
<td>10</td>
<td>(30, 18)</td>
</tr>
</tbody>
</table>

---

Q: Discussion

i. aa Polyline:

---

---

---