Filling Graphics Primitives

September 12, 2008

This note details a few points that the algorithm in the book that could help you in the implementation.

1 Polygon

I would change the `ET`-element and the `AET`-element data structures to contain the following information:

- $y_{\max}$: maximal y-coordinate of the edge
- x: current intersection
- δ: the integer part of $(x_{\max} - x_{\min})/(y_{\max} - y_{\min})$
- ϵ: the remainder part of $(x_{\max} - x_{\min})/(y_{\max} - y_{\min})$
- $dy = y_{\max} - y_{\min}$: the difference between $y_{\max}$ and $y_{\min}$
- cx: the current, accumulated remainder

For example, instead of the `ET`-element $[3, 7, \frac{-5}{2}, next]$, I would store the following element $[3, 7, -2, -1, 2, 0, next]$ where $next$ is the pointer to the next element.

Another example is that the `ET`-element $[5, 7, \frac{6}{4}, next]$ will be stored as $[5, 7, 1, 2, 4, 0, next]$.

1.1 Initializing in `ET`

When we add an edge specified by the two points $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$ to the `AET`, what are the values of δ and ϵ? These two values can be computed as follows:

- $\delta = (x_{\max} - x_{\min})/(y_{\max} - y_{\min})$
- $\epsilon = (x_{\max} - x_{\min})\%(y_{\max} - y_{\min})$

The integer arithmetic of JAVA can be used to determine δ and ϵ.

1.2 Update in `AET`

Given this data structure, I will compute the next intersection using the following algorithm.

Algorithm 1 procedure update(e: `AET`-element)

1: // update current x
2: $e.x := e.x + e.\delta$
3: // update remainder cx
4: $e.cx := e.cx + e.\epsilon$
5: // process cx
6: **if** $|e.cx| \geq |e.dy|$ **then**
7: $\quad e.x := e.x + e.cx/e.dy$
8: $\quad e.cx := e.cx \ \% \ e.dy$
9: **end if**

1.3 Remark

Implicitly, Algorithm 1 does some rounding up in the coordinate of the intersection. In using the value to decide which pixels should be set, the following situations need to be considered:

1. Rounding up when $cx > 0$ — x should be increased by 1
2. Rounding up when $cx < 0$ — nothing needs to be done (already rounded up)
3. Rounding down when $cx > 0$ — nothing needs to be done (already rounded down)
4. Rounding down when $cx < 0$ — x should be decreased by 1

1.4 Example

The next tables demonstrate the `update` algorithm for two edges: AB and BC from Figure 3.22 from the book.

Edge	AB	3	7	-2	-1	2	0
Step in Alg.	Scan line	$y_{\max}$	x	δ	ϵ	dy	cx
	1	3	7	-2	-1	2	0
	2	3	5	-2	-1	2	-1
after line 4	3	3	3	-2	-1	2	-2
final	3	3	2	-2	-1	2	0

Computation for the edge AB

Edge	BC	5	7	1	2	4	0
Step in Alg.	Scan line	$y_{\max}$	x	δ	ϵ	dy	cx
	1	5	7	1	2	4	0
	2	5	8	1	2	4	2
after line 4	3	5	9	1	2	4	4
final	3	5	10	1	2	4	0
	4	5	11	1	2	4	2
after line 4	5	5	12	1	2	4	4
final	5	5	13	1	2	4	0

Computation for the edge BC