Answer to yesterday's exercise

devblogs.microsoft.com/oldnewthing/20030730-00

July 30, 2003

Raymond Chen

iMin is the lowest-index element which intersects the paint rectangle, so a simple truncating division produces the desired index.

The formula for *iMax* can be interpreted two ways. One is that it is the roundup of the first invisible line. Recall the rectangles are exclusive of the endpoint, so *rcPaint.bottom* is actually the first row *outside* the rectangle. Since we want the first element that is completely outside the rectangle, we must round up.

A second interpretation begins with the seemingly equivalent formula. First, the expression

```
    (pps->rcPaint.bottom - 1) / g_cyLine
```

computes the index of the last visible item. By adding unity, we get the index of the first invisible item.

In both cases, we do not allow the computed value to exceed *g_cItems* so we don't try to draw items that don't exist.

The answer to the next question is that the seemingly equivalent formula does not work correctly when *rcPaint.bottom* is zero or negative because the result of integer division is rounded towards zero, which would result in an erroneous computation that the value of *iMax* should be one instead of zero. If integer divisions were rounded towards negative infinity, then the formula would be correct.

And the answer to the final question is that the only harm is that we sometimes draw one item that we really didn't need to. In our example, this is not that big a deal since drawing an item is relatively fast. But in cases where drawing an item is expensive, avoiding the drawing of even a single item may prove significant.

And we'll see in Part 4 that playing with the origin can cause the paint rectangle to end up in odd positions.

Raymond Chen

Follow