USACO Traingate 'window' Analysis

by Alex Schwendner

This problem is very similar to "Shaping Regions". To calculate the visible area of a window, we consider each window. If another window is above the first window, then we split the first window into up to 4 smaller rectangles, (see below) none of which overlaps with the other rectangle, and we recurse on the smaller rectangles.

```
*----------------*
|                |
|                |
|                |
|     *----*     |
|     |    |     |
|     |    |     |   A rectangle partly covers another rectangle.
|     |    |     |
|     *----*     |
|                |
|                |
*----------------*
        ||
        \/
*-----*----*-----*
|     |    |     |
|     | 2  |     |
|     |    |     |
|     *----*     |
|     |    |     |   We remove the covered portion, and divide
|  1  |    |  3  |   the remaining area into 4 rectangles.
|     |    |     |   Of course, this may not always be the layout of
|     *----*     |   the two rectangles; however, we can always use
|     |  4 |     |   this layout if we ignore degenerate rectangles.
|     |    |     |
*-----*----*-----*

       *-----*
       |   2 |
       |     |
*------------------*
|                  |   The same layout works here, we simply ignore rectangles 1
and 3.
|                  |
*------------------*
       |   4 |
       |     |
       *-----*

        *------------*
        |            |
*-------|            |
|       |            |  Here, we ignore rectangles 2 and 3.
|       |            |
|   1   *------------*
|       | 4 |
*-------*---*
```

And here is the solution:

```
#include <fstream.h>
```

```
#include <stdio.h>
#include <assert.h>

template < class type > inline type max (const type & a, const type & b)
{
    return ((a > b) ? a : b);
}

template < class type > inline type min (const type & a, const type & b)
{
    return ((a < b) ? a : b);
}


class   window
{
    public:

    bool real;
    int     y1, x1, y2, x2;             //y1 <= y2, x1 <= x2
    int     level;

    window (void) {
       real = false;
    }

    window (int a, int b, int c, int d, int e) {
       real = true;
       y1 = a;
       x1 = b;
       y2 = c;
       x2 = d;
       level = e;
    }
}

screen[256];

int     top;
int     bot;

inline int
area (window w)
{
    if (w.y1 >= w.y2 || w.x1 >= w.x2) {
       return (0);
    }
    for (int i = 0; i < 256; ++i) {
       if (screen[i].real && screen[i].level > w.level) {
           if (!
              (w.y2 <= screen[i].y1 || screen[i].y2 <= w.y1
                  || w.x2 <= screen[i].x1 || screen[i].x2 <= w.x1)) {
              window  a (w.y1, w.x1, w.y2, screen[i].x1, w.level);
              window  b (w.y1, screen[i].x2, w.y2, w.x2, w.level);
              window  c (w.y1, max (w.x1, screen[i].x1), screen[i].y1,
                          min (screen[i].x2, w.x2), w.level);
              window  d (screen[i].y2, max (w.x1, screen[i].x1), w.y2,
                          min (screen[i].x2, w.x2), w.level);
              return (area (a) + area (b) + area (c) + area (d));
           }
       }
    }
    return ((w.y2 - w.y1) * (w.x2 - w.x1));
```

```
}


//Create window:w (I, x, y, X, Y)
inline void
w (char i, int x1, int y1, int x2, int y2)
{
    assert (!screen[i].real);

    screen[i].real = true;
    screen[i].y1 = y1;
    screen[i].x1 = x1;
    screen[i].y2 = y2;
    screen[i].x2 = x2;

    screen[i].level = top++;
}

//Bring window to top:t (I)
inline void
t (char i)
{
    assert (screen[i].real);
    screen[i].level = top++;
}

//Put window on bottom:b (I)
inline void
b (char i)
{
    assert (screen[i].real);
    screen[i].level = bot--;
}

//Destroy window:d (I)
inline void
d (char i)
{
    assert (screen[i].real);
    screen[i].real = false;
}

//Output percentage visible:s (I)
inline double
s (int i)
{
    assert (screen[i].real);
    return (100.0 * double (area (screen[i])) /
        ((screen[i].y2 - screen[i].y1) * (screen[i].x2 - screen[i].x1)));
}

int
main ()
{
    top = 1;
    bot = 0;

    char    buffer[1000];
    ifstream filein ("window.in");
    FILE *fileout = fopen("window.out", "w");

    while (!filein.eof ()) {
        char    command;
        char    i;
        int     l, m, n, o;
        double  q;

        char    blank;

        filein >> command;
        if (!filein.eof ()) {
            switch (command) {
              case 'w':
                filein >> blank >> i >> blank >> l >> blank >> m >> blank >> n
                    >> blank >> o >> blank;
                assert (!(l == n && m == o));
                w (i, min (l, n), min (m, o), max (l, n), max (m, o));
                break;
              case 't':
                filein >> blank >> i >> blank;
                t (i);
                break;
              case 'b':
                filein >> blank >> i >> blank;
                b (i);
                break;
              case 'd':
                filein >> blank >> i >> blank;
                d (i);
                break;
              case 's':
                filein >> blank >> i >> blank;
                 fprintf(fileout, "%.3f\n", s(i));
                break;
              default:
                cerr << "Bad command \'" << command << "\'.\n";
                return (0);
                break;
            }
        }
    }
    filein.close ();
    fclose (fileout);
    exit (0);
}
```