Subject: Re: Painter bug?
Posted by Mindtraveller on Sat, 12 Mar 2011 21:14:05 GMT
View Forum Message <> Reply to Message

Thank you for the replies. The problem is clear now.
Currently I see the only solution, and I'm not really shure you want it.
Anyway. We could use temporary surface which stores color + alpha for a sequence of operations. On writing a pixel, it's color is calculated in usual way, but alpha is added to the current alpha value. E.g. we will have alpha = 256 in previous Mirek's example. Rendering this temporary surface to actual image will give solid black color which is absolutely right result.
This could look like this:

```
painter
    .BeginComplex()
    ./*draw polygon 1*/
    ./*fill polygon 1*/
    ./*draw polygon 2*/
    ./*fill polygon 2*/
    ./*draw polygon 3*/
    ./*fill polygon 3*/
    .EndComplex()
;
```

UPDATE: OK, here is quick and dirty "back-end" solution. Let's imagine we draw polygons in scaled coordinates:
painter.Scale(scale);
To avoid polygon stitching you may simply move polygon adjacent vertices with 0.5/scale towards neighbouring polygon, i.e.
painter.Move(x[i]+.5/scale,y[i]);
It moves polygon vertex to the next physical pixel if it's position is actually between pixels.
This technique seems like eliminating visual artifacts with U++ rendering of adjacent polygons.