Subject: : AmigaOS4
Topic: : Help me in compiling latest pixman and cairo graphics
Re: Help me in compiling latest pixman and cairo graphics
Author: : Daytona675x
Date: : 2017/9/28 12:48:46
URL:

@AmigaBlitter
Quote:

> Do you know if there's a way to use compositing to accelerate antialiasing?

It depends on what you want to achieve.

You can easily use it to achieve FSAA: just render everything into an offscreeen bitmap which is 2x2 the size than you actually need it to be, scaling all your rendering x2 accordingly.
Then render that offscreen bitmap to the screen using Compositing, scaling it down to your real desired size using filtering, done.

Apart from that, e.g. for single lines there is no way which is free of probably undesired side-effects.

E.g. thellier's approach actually fakes AA by "abusing" bilinear interpolation in combination with fully transparent bitmap areas.
While this may produce sort-of the effect you desire for short diagonal lines, it will look wrong for straight lines (then the filtering will reveal its true nature and appear as some sort of halo around your line).
And depending on your "brush's" shape it will of course have very ugly start / end points unless you take care to compensate the stretching on longer lines by increasing tesselation and only using parts of the bitmap at that areas.

However, something like this is the only way to somehow come close to what you want.
Therefore here are some ideas on how to improve it (not tried, just brainstorming ;) ):

If you want to take that route then your best bet would be to adjust the brush-bitmap and / or UV-coordinates depending on line-size and orientation.
E.g. if you want to draw a straight horizontal line then you should take care that either the brush-bitmap does not contain transparent pixels at the top / bottom or that your V-coordinate is adjusted to not scan those areas during rasterization.

Maybe you can construct an acceptable compromise via some UV-adjustment depending on the line's angle (e.g. if your transparent "border" inside the brush is 3 pixels then the y-border-offset could be something like 3*abs(cos(line_angle*2)). Then add that value to your top-V-coordinate and subtract it from your bottom-V-coordinate.

This would result in:
angle 0 (straight horizontal): 3 (which means to fully remove top/bottom transparent border)
angle 45 (diagonal) : 0 (which means full border and thus full bilinear AA fake-effect)
angle 90 (straight vertical): 3 (like 0)
and so on

Same thing for U of course.

For the start/end "caps" increasing tesselation would probably do, as being said. So instead of 2 triangles render 6, the first fixed size 2 for the line's start, then the stretched main line, then the fixed size end.