
Subject: Re: Strange issue with text in Painter
Posted by [mirek](#) on Mon, 14 Jan 2019 10:11:53 GMT
[View Forum Message](#) <> [Reply to Message](#)

Oh, sorry about the part about "commenting out" - that would not work (because we are using it for the desired rendering as well).

What we need to do is to copy it and THEN start removing...

```
double fx_to_dbl(const FIXED& p);  
Pointf fx_to_dbl(const Pointf& pp, const POINTFX& p);
```

```
struct sMakeGlyph : LRUCache<Value, GlyphKey>::Maker {  
    GlyphKey gk;
```

```
    GlyphKey Key() const    { return gk; }  
    int    Make(Value& v) const {  
        GlyphPainter gp;  
        gp.move = gp.pos = Null;  
        gp.tolerance = gk.tolerance;  
        PaintCharacter(gp, Pointf(0, 0), gk.chr, gk.fnt);  
        int sz = gp.glyph.GetCount() * 4;  
        v = RawPickToValue(pick(gp.glyph));  
        return sz;  
    }  
};
```

```
void RenderCharPath2(const char* gbuf, unsigned total_size, FontGlyphConsumer& sw, double  
xx, double yy)  
{  
    const char* cur_glyph = gbuf;  
    const char* end_glyph = gbuf + total_size;  
    Pointf pp(xx, yy);  
    while(cur_glyph < end_glyph) {  
        const TTPOLYGONHEADER* th = (TTPOLYGONHEADER*)cur_glyph;  
        const char* end_poly = cur_glyph + th->cb;  
        const char* cur_poly = cur_glyph + sizeof(TTPOLYGONHEADER);  
        sw.Move(fx_to_dbl(pp, th->pfxStart));  
        while(cur_poly < end_poly) {  
            const TTPOLYCURVE* pc = (const TTPOLYCURVE*)cur_poly;  
            if (pc->wType == TT_PRIM_LINE)  
                for(int i = 0; i < pc->cpfx; i++)  
                    sw.Line(fx_to_dbl(pp, pc->apfx[i]));  
            if (pc->wType == TT_PRIM_QSPLINE)  
                for(int u = 0; u < pc->cpfx - 1; u++) {  
                    Pointf b = fx_to_dbl(pp, pc->apfx[u]);  
                    Pointf c = fx_to_dbl(pp, pc->apfx[u + 1]);
```

```

    if (u < pc->cpfx - 2)
        c = Mid(b, c);
    sw.Quadratic(b, c);
}
cur_poly += sizeof(WORD) * 2 + sizeof(POINTFX) * pc->cpfx;
}
sw.Close();
cur_glyph += th->cb;
}
}

```

```

HFONT GetWin32Font(Font fnt, int angle);
HDC Win32_IC();

```

```

void RenderCharacterSys2(FontGlyphConsumer& sw, double x, double y, int ch, Font fnt)
{
    HFONT hfont = GetWin32Font(fnt, 0);
    if(hfont) {
        HDC hdc = Win32_IC();
        HFONT ohfont = (HFONT) ::SelectObject(hdc, hfont);
        GLYPHMETRICS gm;
        MAT2 m_matrix;
        memset(&m_matrix, 0, sizeof(m_matrix));
        m_matrix.eM11.value = 1;
        m_matrix.eM22.value = 1;
        int gsz = GetGlyphOutlineW(hdc, ch, GGO_NATIVE, &gm, 0, NULL, &m_matrix);
        if(gsz < 0)
            return;
        StringBuffer gb(gsz);
        gsz = GetGlyphOutlineW(hdc, ch, GGO_NATIVE, &gm, gsz, ~gb, &m_matrix);
        if(gsz < 0)
            return;
        RenderCharPath2(~gb, gsz, sw, x, y + fnt.GetAscent());
        ::SelectObject(hdc, ohfont);
    }
}

```

```

void ApproximateChar(LinearPathConsumer& t, Pointf at, int ch, Font fnt, double tolerance)
{
    PAINTER_TIMING("ApproximateChar");
    Value v;
    INTERLOCKED {
        PAINTER_TIMING("ApproximateChar::Fetch");
        static LRUCache<Value, GlyphKey> cache;
        cache.Shrink(500000);
        sMakeGlyph h;
        h.gk.fnt = fnt;
        h.gk.chr = ch;
    }
}

```

```

h.gk.tolerance = tolerance;
v = cache.Get(h);
#ifdef _DEBUG
DLOG("==== ApproximateChar " << ch << " " << (char)ch << " " << fnt << ", tolerance: " <<
tolerance);
DDUMP(ValueTo< Vector<float> >(v));
GlyphPainter chp;
chp.move = chp.pos = Null;
chp.tolerance = tolerance;

struct PaintCharPath : FontGlyphConsumer {
Painter *sw;

virtual void Move(Pointf p) {
sw->Move(p);
}
virtual void Line(Pointf p) {
sw->Line(p);
}
virtual void Quadratic(Pointf p1, Pointf p2) {
sw->Quadratic(p1, p2);
}
virtual void Cubic(Pointf p1, Pointf p2, Pointf p3) {
sw->Cubic(p1, p2, p3);
}
virtual void Close() {
sw->Close();
}
} pw;
pw.sw = &chp;
RenderCharacterSys2(pw, 0, 0, ch, fnt);
// fnt.Render(pw, 0, 0, ch);
// PaintCharacter(chp, Pointf(0, 0), ch, fnt);
// DDUMP(chp.glyph);
// ASSERT(ValueTo< Vector<float> >(v) == chp.glyph);
#endif
}
const Vector<float>& g = ValueTo< Vector<float> >(v);
int i = 0;
while(i < g.GetCount()) {
Pointf p;
p.x = g[i++];
if(p.x > 1e30) {
p.x = g[i++];
p.y = g[i++];
t.Move(p + at);
}
else {

```

```
PAINTER_TIMING("ApproximateChar::Line");
p.y = g[i++];
t.Line(p + at);
}
}
}
```

It this still 'toggles', start commenting from RenderCharPath2 call up...
