Subject: Re: Strange issue with text in Painter
Posted by mirek on Mon, 14 Jan 2019 14:03:30 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 14 January 2019 14:11Not a single false character in DEBUG mode either. :)

I should have found something already if the problem still existed.

So, is this the fix or means for problem isolation?

Best regards,

Tom

This basically disables MT for character rendering...

So for now, please revert it and try two separate things for me:

```
void Painter::TextOp(const Pointf& p, const wchar *text, Font fnt, int n, const double *dx)
{
 if(n == 0) {
  Move(0, 0);
  return;
 }
 FontInfo fi = fnt.Info();
 double x = p.x;
 while(n) {
  int ch = *text++;
  Character(x, p.y, ch, fnt);
//  Div();
  if(dx)
   x += *dx++;
  else
   x += fi[ch];
  n--;
 }
 if(fnt.IsUnderline() || fnt.IsStrikeout()) {
  double a = fnt.GetAscent();
  double cy = max(a / 16, 1.0);
  double cx = x - p.x;
  if(fnt.IsUnderline())
   Rectangle(p.x, p.y + a + cy, cx, cy);
  if(fnt.IsStrikeout())
   Rectangle(p.x, p.y + 2 * a / 3, cx, cy);
 }
}
```

(This might lead to some new artifacts where characters overlap, that is expected, look for the original problem (wrong characters).

```
void BufferPainter::RenderPathSegments(LinearPathConsumer *g, const Vector<byte>& path,
                                       const SimpleAttr *attr, double tolerance)
{
 Pointf pos = Pointf(0, 0);
 const byte *data = path.begin();
 const byte *end = path.end();
 while(data < end) {
  const LinearData *d = (LinearData *)data;
  switch(d->type) {
  case MOVE: {
   g->Move(pos = attr ? attr->mtx.Transform(d->p) : d->p);
   data += sizeof(LinearData);
   break;
  }
  case LINE: {
   PAINTER_TIMING("LINE");
   g->Line(pos = attr ? attr->mtx.Transform(d->p) : d->p);
   data += sizeof(LinearData);
   break;
  }
  case QUADRATIC: {
   PAINTER_TIMING("QUADRATIC");
   const QuadraticData *d = (QuadraticData *)data;
   if(attr) {
    Pointf p = attr->mtx.Transform(d->p);
    ApproximateQuadratic(*g, pos, attr->mtx.Transform(d->p1), p, tolerance);
    pos = p;
   }
   else {
    ApproximateQuadratic(*g, pos, d->p1, d->p, tolerance);
    pos = d->p;
   }
   data += sizeof(QuadraticData);
   break;
  }
  case CUBIC: {
   PAINTER_TIMING("CUBIC");
   const CubicData *d = (CubicData *)data;
   if(attr) {
    Pointf p = attr->mtx.Transform(d->p);
    ApproximateCubic(*g, pos, attr->mtx.Transform(d->p1),
                     attr->mtx.Transform(d->p2), p, tolerance);
    pos = p;
   }
   else {
    ApproximateCubic(*g, pos, d->p1, d->p2, d->p, tolerance);
    pos = d->p;
   }
   data += sizeof(CubicData);
   break;
  }
  case CHAR: {
   const CharData *ch = (CharData *)data;
   INTERLOCKED ApproximateChar(*g, ch->p, ch->ch, ch->fnt, tolerance);
   data += sizeof(CharData);
   break;
  }
  default:
   NEVER();
   g->End();
   return;
  }
 }
 g->End();
}
```

(Additional mutex in for CHAR).

Thanks,

Mirek