Subject: Re: BufferPainter::Fill(Image,...) optimization question
Posted by mirek on Thu, 04 Jun 2020 16:38:30 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 27 April 2020 16:51Hi,

Never mind... after all I managed to get through and found the fillers.

While this is not much, I noticed that the following changes improve BufferPainter::Fill(Image,...) performance by about 5 % for MT and about 13 % for ST on my computer. The changes (centered around adding and handling of 'kind' for SpanSource) follow.

Painter/BufferPainter.h:

```
struct SpanSource {
 int kind;
 SpanSource(){
  kind = IMAGE_OPAQUE;
 }

 virtual void Get(RGBA *span, int x, int y, unsigned len) = 0;
 virtual ~SpanSource() {}
};
```

Painter/Fillers.cpp:

```
void SpanFiller::Render(int val, int len)
{
 if(val == 0) {
  t += len;
  s += len;
  return;
 }
 if(alpha != 256)
  val = alpha * val >> 8;

 if(val == 256) {
  if(ss->kind==IMAGE_OPAQUE) memcpy(t,s,len*sizeof(RGBA));
  else{
   for(int i = 0; i < len; i++) {
    if(s[i].a == 255)
     t[i] = s[i];
    else
     AlphaBlend(t[i], s[i]);
   }
  }
  t += len;
  s += len;
 }
 else {
```

```
  const RGBA *e = t + len;
  while(t < e)
   AlphaBlendCover8(*t++, *s++, val);
 }
}
```

Painter/Image.cpp:

```
struct PainterImageSpan : SpanSource, PainterImageSpanData {
 LinearInterpolator interpolator;

 PainterImageSpan(const PainterImageSpanData& f)
 : PainterImageSpanData(f) {
  interpolator.Set(xform);
  kind = image.GetKindNoScan(); // Tom added
 }
```

This just leaves me wondering why is the improvement so insignificant, no matter there is no longer any comparison and/or blending required. Is there yet another layer of transferring pixels somewhere?

Please review the changes. If they are correct and sensible -- which I'm not sure about -- feel free to merge.

Best regards,

Tom

EDIT: Changed default SpanSource::kind to IMAGE_OPAQUE to boost all kinds of filling. The change introduced a slight improvement over the previous round.

I have moved over to this, unfortunately it is more complicated because even opaque image can return zero alpha for areas it does not cover....
