Subject: BufferPainter::Fill(Image,...) optimization question
Posted by Tom1 on Mon, 27 Apr 2020 12:51:48 GMT
View Forum Message <> Reply to Message

Hi Mirek,

After getting inspired by the recent improvements in DrawImage(), I tested the BufferPainter::Fill(Image,...) using images with both kind=IMAGE_OPAQUE and kind=IMAGE_ALPHA. It seems they apparently render at the same speed. Is there any room for "easy" optimization when rendering opaque images?

(I really tried to look for the code where the rendering is actually done, but BufferPainter is way too complex for me to navigate... :? Maybe you could point out the file and line where the decision between opaque rendering and alpha blending is actually done.)

Best regards,

Tom

Subject: Re: BufferPainter::Fill(Image,...) optimization question
Posted by Tom1 on Mon, 27 Apr 2020 14:51:01 GMT
View Forum Message <> Reply to Message

Hi,

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.

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....

Subject: Re: BufferPainter::Fill(Image,...) optimization question
Posted by mirek on Thu, 04 Jun 2020 16:49:42 GMT
View Forum Message <> Reply to Message

So this is (likely) the correct code:

```
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 && ss->opaque) {
  const RGBA *e = s + len;
  const RGBA *s0 = s;
  while(s < e && s->a == 0)
   s++;
  t += s - s0;
  s0 = s;
  while(s < e && s->a != 255)
   s++;
  AlphaBlend(t, s0, val, s - s0);
  t += s - s0;
  s0 = s;
  while(s < e && s->a == 255)
   s++;
  memcpy32(t, s0, s - s0);
  t += s - s0;
  s0 = s;
  while(s < e && s->a)
   s++;
  AlphaBlend(t, s0, val, s - s0);
 }
 else
  AlphaBlend(t, s, val, len);
 t += len;
 s += len;
}
```

Unfortunately it is then slower than new SSE2 AlphaBlend path, so...

Mirek

Subject: Re: BufferPainter::Fill(Image,...) optimization question
Posted by Tom1 on Thu, 04 Jun 2020 18:17:59 GMT
View Forum Message <> Reply to Message

Hi,

Yes, I noticed that the AlphaBlend optimization improved this opaque image (FAST_FILL) speed by about 20 %, so it covered this area too. :)

Thanks and best regards,

Tom