Subject: Painter Fill with Image MSC14x64 performance issue
Posted by Tom1 on Tue, 10 Jan 2017 13:17:06 GMT
View Forum Message <> Reply to Message

Hi Mirek,

In your free time (does it even exist?) could you take a look at this?

Using Fill in e.g. BufferPainter to draw an Image (with FAST_FILL option) performs slower than expected when compiled with MSC14x64 compared to e.g. MSC14. For example, a 256x256 pixel Image zoomed in to cover the whole UHD display resolution takes about 97 ms to render when compiled with MSC14x64, whereas the same only takes 59 ms for MSC14. On Linux, the typical performance figures are between 50 ms (CLANG) and 60 ms (GCC), but these are measured on a VirtualBox virtual machine on top of my Windows 10, so I'm not 100 % confident about these Linux numbers.

Mostly I have experienced small to moderate speed improvements when switching to 64-bit, but now it's the other way around and with quite a big marginal too.

Is there a way to make the code more x64 friendly to squeeze more speed out of it?

PainterExamples demonstrates this issue when using fast_fill for an image rendered at e.g. 3x.

(I do not know if this has any significance for the subject, but my CPU is an Intel i7 and OS is Windows 10 Professional 64-bit.)

If you are way too busy to look at this, could you point to the place in Painter package where I should start looking at?

Best regards,

Tom

Subject: Re: Painter Fill with Image MSC14x64 performance issue
Posted by mirek on Tue, 10 Jan 2017 16:48:39 GMT
View Forum Message <> Reply to Message

Interesting. I would start digging around

struct PainterImageSpan

Mirek

Subject: Re: Painter Fill with Image MSC14x64 performance issue
Posted by Tom1 on Wed, 11 Jan 2017 09:01:32 GMT

Hi Mirek,

Thanks for pointing out the correct location for investigation:

With a few changes, I have succeeded to more than double the performance: UHD sized FAST_FILL render was dropped from 97 ms to 35 ms when using MSC14x64. For MSC14 the improvement was observable with a drop from 59 ms to 40 ms.

First BufferPainter.h in class LinearInterpolator I have inlined 'int Dda2::Get()' and 'Point Get()' functions (and removed them from Interpolator.cpp):

```
class LinearInterpolator {
 struct Dda2 {
  int count, lift, rem, mod, p;

  void  Set(int a, int b, int len);
//  int   Get();
  int Get()
  {
   int pp = p;
   mod += rem;
   p += lift;
   if(mod > 0) {
    mod -= count;
    p++;
   }
   return pp;
  }

 };

 Xform2D xform;
 Dda2    ddax, dday;

 static int Q8(double x) { return int(256 * x + 0.5); }

public:
 void   Set(const Xform2D& m)                  { xform = m; }

 void   Begin(int x, int y, int len);
// Point  Get();
 Point Get()
 {
  return Point(ddax.Get(), dday.Get());
 }

};
```

Second Image.cpp in struct PainterImageSpan at the beginning of 'virtual void Get(RGBA *span, int x, int y, unsigned len)' I have added optimized code for FAST_FILL without effect flags. This mostly improves MSC14x64 results:

```
 virtual void Get(RGBA *span, int x, int y, unsigned len)
 {
  interpolator.Begin(x, y, len);
  fixed = hstyle && vstyle;

  if((hstyle|vstyle)==0 && fast){
   while(len--){
    Point l = interpolator.Get() >> 8;
    if(l.x > 0 && l.x < maxx && l.y > 0 && l.y < maxy) *span = Pixel(l.x, l.y);
    else if(style == 0 && (l.x < -1 || l.x > cx || l.y < -1 || l.y > cy)) *span = RGBAZero();
    else *span = GetPixel(l.x, l.y);
    ++span;
   }
   return;
  }

  while(len--) {
   Point h = interpolator.Get();
 //  h -= 128;
   Point l = h >> 8;
 ...
```

Finally, some more milliseconds can be squeezed out by changing SpanFiller::Render(int val, int len) in Fillers.cpp as follows. Using 'for' instead of 'while' seems to have positive effect mostly on MSC14x64:

```
void SpanFiller::Render(int val, int len)
{
 if(val == 0) {
  t += len;
  s += len;
  return;
 }
 const RGBA *e = t + len;
 if(alpha != 256)
  val = alpha * val >> 8;
 if(val == 256)
  for(int i=0;i<len;i++) if(s[i].a==255) t[i]=s[i]; else AlphaBlend(t[i], s[i]);
/*  while(t < e) {
   if(s->a == 255)
    *t++ = *s++;
   else
    AlphaBlend(*t++, *s++);
```

```
  }
*/ else
  while(t < e)
   AlphaBlendCover8(*t++, *s++, val);
}
```

Please evaluate the changes and commit if you agree with me.

Best regards,

Tom

UPDATE: There was a slight error in the positioning of "fixed = hstyle && vstyle;" potentially causing crash. Fixed above.

Another UPDATE: There was an extra "LinearInterpolator::" for "Point LinearInterpolator::Get()" above. For Windows it was ok, but CLANG found it. Sorry for that one too. Fixed now.

Subject: Re: Painter Fill with Image MSC14x64 performance issue
Posted by mirek on Wed, 11 Jan 2017 19:39:26 GMT
View Forum Message <> Reply to Message

Good work. Some of these are unexpected, but make sense. Applied (with some cosmetic changes).

Subject: Re: Painter Fill with Image MSC14x64 performance issue
Posted by koldo on Fri, 13 Jan 2017 07:47:51 GMT
View Forum Message <> Reply to Message

Yes, good work. Thank you.

Subject: Re: Painter Fill with Image MSC14x64 performance issue
Posted by mirek on Sun, 29 Jan 2017 20:04:07 GMT
View Forum Message <> Reply to Message

Quote:

```
void SpanFiller::Render(int val, int len)
{
 if(val == 0) {
  t += len;
  s += len;
  return;
```

```
 }
 const RGBA *e = t + len;
 if(alpha != 256)
  val = alpha * val >> 8;
 if(val == 256)
  for(int i=0;i<len;i++) if(s[i].a==255) t[i]=s[i]; else AlphaBlend(t[i], s[i]);
/*  while(t < e) {
   if(s->a == 255)
    *t++ = *s++;
   else
    AlphaBlend(*t++, *s++);
  }
*/ else
  while(t < e)
   AlphaBlendCover8(*t++, *s++, val);
}
```

Missed ugly bug above: t and s are member variables and need to be moved after the loop:

```
void SpanFiller::Render(int val, int len)
{
 if(val == 0) {
  t += len;
  s += len;
  return;
 }
 const RGBA *e = t + len;
 if(alpha != 256)
  val = alpha * val >> 8;
 if(val == 256) {
  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
  while(t < e)
   AlphaBlendCover8(*t++, *s++, val);
}
```

[/quote]

Maybe you could check whether this correct code is still faster?

Mirek

Subject: Re: Painter Fill with Image MSC14x64 performance issue
Posted by Tom1 on Mon, 30 Jan 2017 07:35:48 GMT
View Forum Message <> Reply to Message

Hi,

Adding the:

```
t += len;
s += len;
```

introduces nearly undetectable penalty on my tests. Sorry for my mistake.

Best regards,

Tom

Subject: Re: Painter Fill with Image MSC14x64 performance issue
Posted by Tom1 on Mon, 30 Jan 2017 08:16:33 GMT
View Forum Message <> Reply to Message

As a matter of fact, this is even better:

```
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) {
  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);
 }
}
```

I.e. moving "const RGBA *e = t + len;" to the else section where it is actually needed. (A very slight but measurable speed improvement can be observed.)

Best regards,

Tom
