Subject: Re: BufferPainter::Clear() optimization
Posted by mirek on Tue, 19 May 2020 09:32:39 GMT
View Forum Message <> Reply to Message

Three more variants, based on your FillT. Fill7 is basically identical, with a little trick added (hope you like it). Fill7a has different "frontend". Fill8 is not performing very well, adding that just so that you know I have tested that variant too... :)

Fill7 and Fill7a seem to be basically equal and maybe just a tiny bit faster than Fill3T....

```
void Fill7(dword *t, dword data, int len){
 switch(len) {
  case 3: t[2] = data;
  case 2: t[1] = data;
  case 1: t[0] = data;
  case 0: return;
 }

 __m128i val4 = _mm_set1_epi32(data);
 auto Set4 = [&](int at) { _mm_storeu_si128((__m128i *)(t + at), val4); };

 Set4(len - 4); // fill tail
 if(len >= 32) {
  if(len >= 1024*1024) { // for really huge data, bypass the cache
   HugeFill(t, data, len);
   return;
  }
  const dword *e = t + len - 32;
  do {
   Set4(0); Set4(4); Set4(8); Set4(12);
   Set4(16); Set4(20); Set4(24); Set4(28);
   t += 32;
  }
  while(t <= e);
 }
 if(len & 16) {
  Set4(0); Set4(4); Set4(8); Set4(12);
  t += 16;
 }
 if(len & 8) {
  Set4(0); Set4(4);
  t += 8;
 }
 if(len & 4)
  Set4(0);
}
```

```
void Fill7a(dword *t, dword data, int len){
 if(len < 4) {
  if(len & 2) {
   t[0] = t[1] = data;
   t += 2;
  }
  if(len & 1)
   t[0] = data;
  return;
 }

 __m128i val4 = _mm_set1_epi32(data);
 auto Set4 = [&](int at) { _mm_storeu_si128((__m128i *)(t + at), val4); };

 Set4(len - 4); // fill tail
 if(len >= 32) {
  if(len >= 1024*1024) { // for really huge data, bypass the cache
   HugeFill(t, data, len);
   return;
  }
  const dword *e = t + len - 32;
  do {
   Set4(0); Set4(4); Set4(8); Set4(12);
   Set4(16); Set4(20); Set4(24); Set4(28);
   t += 32;
  }
  while(t <= e);
 }
 if(len & 16) {
  Set4(0); Set4(4); Set4(8); Set4(12);
  t += 16;
 }
 if(len & 8) {
  Set4(0); Set4(4);
  t += 8;
 }
 if(len & 4)
  Set4(0);
}

void Fill8(dword *t, dword data, int len){
 switch(len) {
  case 3: t[2] = data;
  case 2: t[1] = data;
  case 1: t[0] = data;
  case 0: return;
 }
```

```
 __m128i val4 = _mm_set1_epi32(data);
 auto Set4 = [&](int at) { _mm_storeu_si128((__m128i *)(t + at), val4); };

 Set4(len - 4); // fill tail
 if(len >= 32) {
  if(len >= 1024*1024) { // for really huge data, bypass the cache
   HugeFill(t, data, len);
   return;
  }
  int cnt = len >> 5;
  do {
   Set4(0); Set4(4); Set4(8); Set4(12);
   len -= 32;
   Set4(16); Set4(20); Set4(24); Set4(28);
   t += 32;
  }
  while(len >= 32);
 }
 switch((len >> 2) & 7) {
 case 7: Set4(24);
 case 6: Set4(20);
 case 5: Set4(16);
 case 4: Set4(12);
 case 3: Set4(8);
 case 2: Set4(4);
 case 1: Set4(0);
 }
}
```