Subject: Re: BufferPainter::Clear() optimization
Posted by mirek on Fri, 15 May 2020 11:15:51 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 15 May 2020 12:08
Additionally, plain memset, memsets and memsetd -variants would be useful for various tasks, as their efficiency varies depending on the compiler.

What about this:

```
void FillCacheLines(void *cache_aligned_ptr, void *data16, int count)
{
 dword *t = (dword *)cache_aligned_ptr;
 __m128d val = _mm_loadu_pd((double *)data16);
 dword *e = t + 16 * count;
 while(t < e) {
  _mm_stream_pd((double *)t, val);
  _mm_stream_pd((double *)(t + 4), val);
  _mm_stream_pd((double *)(t + 8), val);
  _mm_stream_pd((double *)(t + 12), val);
  t += 16;
 }
 _mm_sfence();
}

template <class T>
void MemSet(void *dest, T data, int len)
{
 static_assert(sizeof(T) == 1 || sizeof(T) == 2 || sizeof(T) == 4 || sizeof(T) == 8 || sizeof(T) == 16,
"invalid sizeof");
 T *t = (T *)dest;
 if(len * sizeof(T) > 550) {
  while((uintptr_t)t & 63) { // align to cache line
   *t++ = data;
   len--;
  }
  const int itemn = 16 / sizeof(T);
  const int per_cache_line = 4 * itemn;
  T m[itemn];
  for(int i = 0; i < itemn; i++)
   m[i] = data;
  int count = len / per_cache_line;
  FillCacheLines(t, m, count);
  len -= per_cache_line * count;
 }
```

```
 while(len >= 16) {
  t[0] = data; t[1] = data; t[2] = data; t[3] = data;
  t[4] = data; t[5] = data; t[6] = data; t[7] = data;
  t[8] = data; t[9] = data; t[10] = data; t[11] = data;
  t[12] = data; t[13] = data; t[14] = data; t[15] = data;
  t += 16;
  len -= 16;
 }
 switch(len) {
 case 15: t[14] = data;
 case 14: t[13] = data;
 case 13: t[12] = data;
 case 12: t[11] = data;
 case 11: t[10] = data;
 case 10: t[9] = data;
 case 9: t[8] = data;
 case 8: t[7] = data;
 case 7: t[6] = data;
 case 6: t[5] = data;
 case 5: t[4] = data;
 case 4: t[3] = data;
 case 3: t[2] = data;
 case 2: t[1] = data;
 case 1: t[0] = data;
 }
}
```