Subject: Re: Painter refactored/optimized
Posted by Tom1 on Fri, 16 Nov 2018 09:23:09 GMT
View Forum Message <> Reply to Message

Hi Mirek,

While on the subject, I decided to do some testing of thread count for MT Painter. What I found was interesting: My typical map renders at roughly 250 ms with ST and 100 ms with default 10 thread MT. (On my hardware CPU_Cores() returns 8 and CoWork initializes a thread pool of 10 threads.)

So I tampered a little bit with CoWork.cpp, trying with different thread counts:

```
int CoWork::GetPoolSize()
{
 int n = GetPool().threads.GetCount();
// return n ? n : CPU_Cores() + 2;
 return n ? n : 4;
}

CoWork::Pool::Pool()
{
 ASSERT(!IsWorker());

// InitThreads(CPU_Cores() + 2);
 InitThreads(4);

 free = NULL;
 for(int i = 0; i < SCHEDULED_MAX; i++)
  Free(slot[i]);

 quit = false;
}
```

In this test I ended up with four threads which yield about same performance as 10 threads. When dropping to three threads or below, the MT gain started to fade away.

I think the optimal thread count for CoWork depends on the job's balance of CPU load and memory bandwidth. Also, the CPU and memory bus design changes this balance. As the new CPUs tend to offer a lot of cores (and concurrent threads), a simple or well optimized algorithm will easily saturate the memory channels with a reasonably small subset of cores being used. I'm not sure though, if there is much point in reducing threads (and therefore freeing cores for other tasks), if the memory bus will remain saturated anyway.

Best regards,

Tom