Subject: High resolution TimeStop code
Posted by cbpporter on Thu, 05 Mar 2015 10:30:37 GMT
View Forum Message <> Reply to Message

As mentioned in another thread, I require high resolution timers. Most things that one need to measure require precise measurements and TimeStop has a minimum time increment of 0/15/16 msec.

Here is a code I rewrote (I keep loosing ti each time I change U++ version) for TimeStopHR, which has adequate but not maximum precision:

```
class TimeStopHR : Moveable<TimeStopHR> {
 LARGE_INTEGER start;
  LARGE_INTEGER stop;
  LARGE_INTEGER freq;

public:
 double  Elapsed() {
  QueryPerformanceCounter(&stop);
  return (stop.QuadPart - start.QuadPart) * 1000.0 / freq.QuadPart;
 }

 double Seconds()                { return (double)Elapsed() / 1000; }
 String ToString();
 void   Reset();

 TimeStopHR();

 static void SetProcessorAffinity();
};



void TimeStopHR::Reset()
{
 QueryPerformanceFrequency(&freq);
 QueryPerformanceCounter(&start);
}

TimeStopHR::TimeStopHR()
{
 Reset();
}

String TimeStopHR::ToString()
{
 double time = Elapsed();
```

```
 return Format("%d.%03d", int(time / 1000), int(time) % 1000);
}

void TimeStopHR::SetProcessorAffinity()
{
    // Assign the current thread to one processor. This ensures that timing
    // code runs on only one processor, and will not suffer any ill effects
    // from power management.
    //
    // Based on the DXUTSetProcessorAffinity() function in the DXUT framework.

    DWORD_PTR dwProcessAffinityMask = 0;
    DWORD_PTR dwSystemAffinityMask = 0;
    HANDLE hCurrentProcess = GetCurrentProcess();

    if (!GetProcessAffinityMask(hCurrentProcess, &dwProcessAffinityMask,
&dwSystemAffinityMask))
        return;

    if (dwProcessAffinityMask)
    {
        // Find the lowest processor that our process is allowed to run against.

        DWORD_PTR dwAffinityMask = (dwProcessAffinityMask & ((~dwProcessAffinityMask) + 1));

        // Set this as the processor that our thread must always run against.
        // This must be a subset of the process affinity mask.

        HANDLE hCurrentThread = GetCurrentThread();

        if (hCurrentThread != INVALID_HANDLE_VALUE)
        {
            SetThreadAffinityMask(hCurrentThread, dwAffinityMask);
            CloseHandle(hCurrentThread);
        }
    }

    CloseHandle(hCurrentProcess);
}
```

SetProcessorAffinity is optional and ensures even more added precision.

Could we get this added to core as a new class or a replacement for the GetTickCount based TimeStop?