

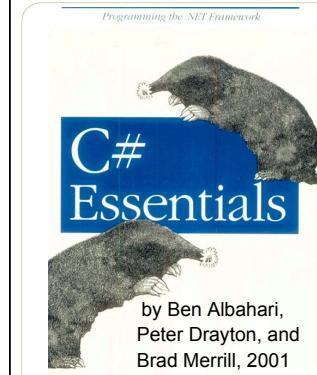
**ECE4893A/CS4803MPG:
MULTICORE AND GPU
PROGRAMMING
FOR VIDEO GAMES**

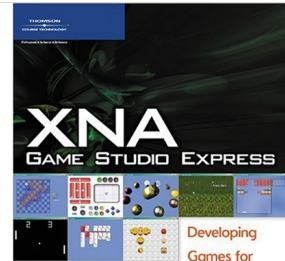
 **Pulse/Wait and Semaphores** 

Prof. Aaron Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology



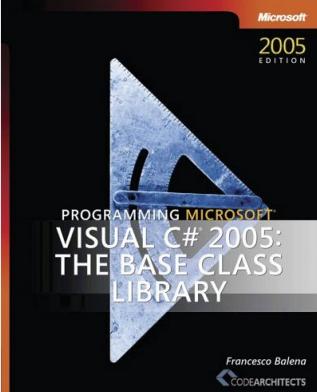
References (1)


by Ben Albahari,
Peter Drayton, and
Brad Merrill, 2001


Developing
Games for
Windows and
the Xbox 360
by Joseph Hall, 2008



Reference (2)


Francesco
Balena
2006
Microsoft
Press



Pulse and wait

```
using System;
using System.Threading;
class MonitorTest {
    static void Main() {
        MonitorTest mt = new MonitorTest();
        Thread t = new Thread(new ThreadStart(mt.Go));
        t.Start();
        mt.Go();
    }
    void Go() {
        for (char c='a'; c <= 'z'; c++)
            lock(this) {
                Console.Write(c);
                Monitor.Pulse(this);      // wake up next thread that
                Monitor.Wait(this);       // is waiting on the object
            }                         // once I've released it
    }
}
```

↑ release lock temporarily; go to sleep until another thread pulses me

Example from "C# Essentials," p. 109



Pulse and wait example output

```
using System;
using System.Threading;
class MonitorTest {
    static void Main() {
        MonitorTest mt = new MonitorTest();
        Thread t = new Thread(new ThreadStart(mt.Go));
        t.Start();
        mt.Go();
    }
    void Go() {
        for (char c='a'; c <= 'z'; c++)
            lock(this) {
                Console.Write(c);
                Monitor.Pulse(this);
                Monitor.Wait(this);
            }
    }
}
```

Output:

aabbccddeeffgghhiijjkkllmm
nnooppqqrrssttuuvvwwwxxyyzz

Example from
“C#
Essentials,”
p. 108



5

What's the problem?

```
using System;
using System.Threading;
class MonitorTest {
    static void Main() {
        MonitorTest mt = new MonitorTest();
        Thread t = new Thread(new ThreadStart(mt.Go));
        t.Start();
        mt.Go();
    }
    void Go() {
        for (char c='a'; c <= 'z'; c++)
            lock(this) {
                Console.Write(c);
                Monitor.Pulse(this);
                Monitor.Wait(this);
            }
    }
}
```

Example from
“C#
Essentials,”
p. 109



6

Breaking the deadlock

```
void Go() {
    for (char c='a'; c <= 'z'; c++)
        lock(this) {
            Console.Write(c);
            Monitor.Pulse(this);
            if (c < 'z')
                Monitor.Wait(this);
        }
}
```

Example from “C# Essentials,” p. 110



7

Impatient Wait

```
public static bool Wait(object obj,
                        int millisecondsTimeout);
```

- If another thread doesn't pulse me within millisecondsTimeout, reacquire the lock and wake myself up
- Return true if reactivated by monitor being pulsed
- Return false if wait timed out



8

Grrrrrrrrrrrr!!!!

- XNA on Xbox 360 uses Compact Framework, not full .NET like on Windows
- Compact Framework has a Monitor class (so can use locks), but it doesn't implement Pulse/Wait and their variations ☹
- Not sure about "pro Xbox 360 development," i.e. C++ XDK

 Georgia Institute
of Technolgy

9

Semaphores

- Semaphores are good for restricting the number of threads accessing a resource to some maximum number
- As far as I can tell, in XNA, semaphores are only available on Windows ☹
 - Seems to be missing in Compact Framework and hence XNA on Xbox 360
 - Rumor has it semaphores are available in "pro Xbox 360 development," i.e. C++ XDK

 Georgia Institute
of Technolgy

10

Semaphores in C# (1)

```
Semaphore sem = new Semaphore(2, 2);
                ↑   ↑
                Initial count Max count

sem.WaitOne(); // count from 2 to 1
sem.Release(); // count from 1 to 2
sem.Release(); // tries to bring
               // count from 2 to 3,
               // but throws a
               // SemaphoreFull
               // exception
```

From F. Balena, "Visual C# 2005: The Base Class Library," p. 482.

 Georgia Institute
of Technolgy

11

Semaphores in C# (2)

```
Semaphore sem = new Semaphore(2, 2);
                ↑   ↑
                Initial count Max count

sem.WaitOne(); // count from 2 to 1
sem.WaitOne(); // count from 1 to 0
sem.WaitOne(); // Blocks until
               // another thread
               // calls sem.Release();
```

From F. Balena, "Visual C# 2005: The Base Class Library," p. 482.

 Georgia Institute
of Technolgy

12

Typical use of a semaphore

```
Semaphore sem = new Semaphore(2,2);

void SemaphoreExample()
{
    // Wait until a resource becomes available.
    sem.WaitOne();
    // Enter the synchronized section

    // Exit the synchronized section, and
    // release the resource
    sem.Release();
}
```

From F. Balena, "Visual C# 2005: The Base Class Library," p. 478.



13

Semaphores: naming and timeouts

- Semaphores can be named like mutexes (makes sense on Windows)
- Like event waits (pulse/wait from previous lecture), semaphore and mutex waits can be given timeout parameter, and return a boolean indicating whether they acquired the resource “naturally” or timed out



14

Semaphore with max count 1 vs. Mutex

- A mutex or Monitor lock is owned by a thread; only that thread can release it
- Semaphores can be released by anyone



15