ECE4893A/CS4803MPG: Multicore and GPU Programming for Video Games

Lecture 13: Introduction to Multithreading

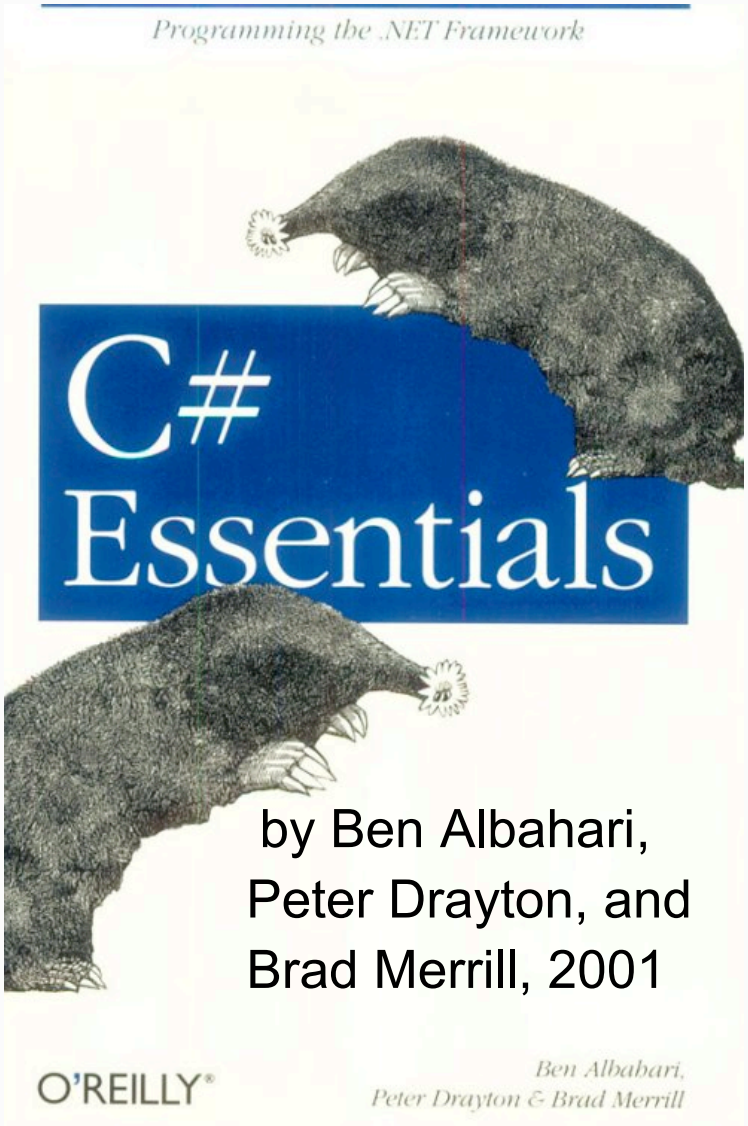
Prof. Aaron Lanterman

School of Electrical and Computer Engineering

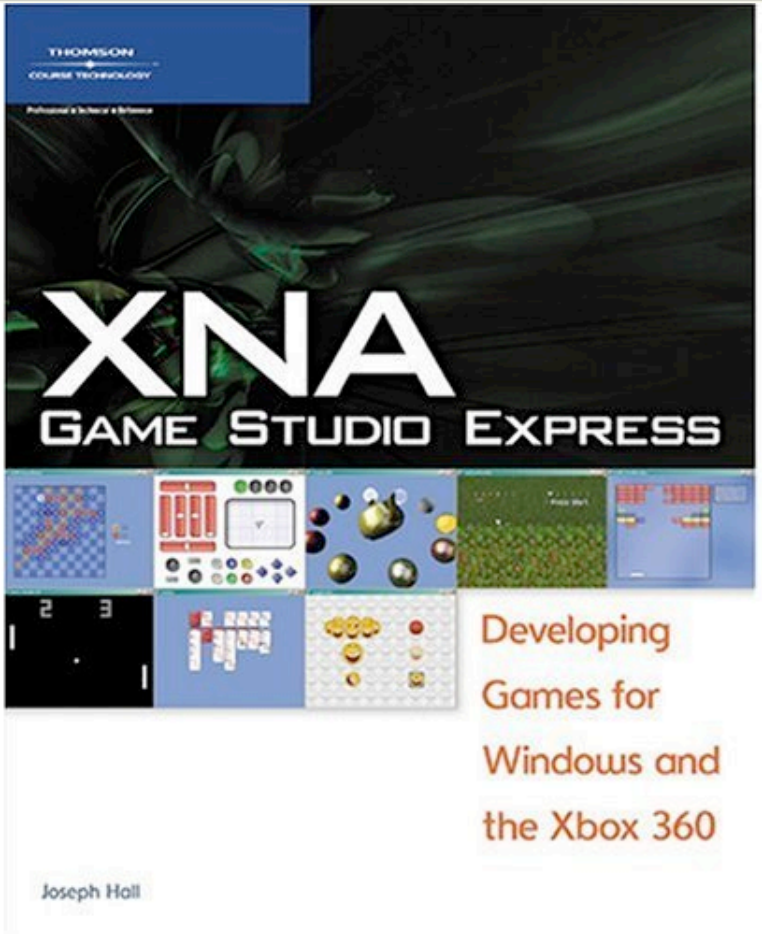
Georgia Institute of Technology

Georgia Institute of Technology

References



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



by Joseph Hall, 2008

Georgia Institute of Technology

Threading example

```
using System;
using System.Threading;
class ThreadTest
{
  static void Main()
  {
    Thread t = new Thread(new ThreadStart(Go));
    t.Start();
    Go();
  }
  static void Go()
  {
    for (char c='a'; c <= 'z'; c++)
      Console.Write(c);
  }
}
```

static methods are part of the class, not particular instances

Example from "C# Essentials," pp. 107-108.

Georgia Institute of Technology

Threading example output

```
using System
using System.Threading;
class ThreadTest
{
  static void Main()
  {
     Thread t = new Thread(new ThreadStart(Go));
    t.Start();
    Go();
   }
  static void Go()
  {
      for (char c='a'; c <= 'z'; c++)
      Console.Write(c);
  }
}
```

Output:

```
abcdabcdefghijklmnopqrsefg
hjiklmnopqrstuvwxyztuvwxyz
```

Example from "C# Essentials," pp. 107-108.

Georgia Institute of Technology

Lock example

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

Example from "C# Essentials," p. 108

this references the current instance of the class (can't use **this** in static methods)

lock takes a reference type; if another thread has already acquired a lock, this thread halts until the other thread lets it go

Locks example output

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

Example from "C# Essentials," p. 108

Output:

```
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

Georgia Institute of Technology

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);
        Monitor.Wait(this);
      }
  }
}
```

Example from "C# Essentials," p. 109

wake up next thread that is waiting on the object once I've released it

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

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);
      }
  }
}
```

Example from "C# Essentials," p. 108

Output:

```
aabbccddeeffgghhiijjkkllmm
nnooppqqrrssttuuvvwwxxyyzz
```

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

wake up next thread that is waiting on the object once I've released it

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

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

Lock: behind the curtain

```
lock(expression)
{
  //mycode
}
```

is syntactic sugar for

```
System.Threading.Monitor.Enter(expression);
try {
  // mycode
}
finally {
  System.Threading.Monitor.Exit(expression);
}
```

From “C# Essentials,” pp. 108-109

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

Polling

- Main thread checks flag variables set by the worker threads when they finish
- Useful if main thread can do some stuff (e.g., eye-candy animation in a turn-based strategy game) independently of the worker threads (e.g. AI), but needs worker threads to finish before continuing (e.g. making the computer's move)

Polling example

Code from Joseph Hall, "XNA Game Studio Express," p. 608

```
bool done = false;
while (!done)
{
  Thread.Sleep(0);
  done = true;
  for int(i = 0; i < ThreadDone.Length;
                                    i++)
  {
    done &= m_ThreadDone[i];
  }
}
```

Worker thread i sets `m_ThreadDone[i]=true` before it exits

The problem with polling

- Polling takes up “C# cycles”
- If your main thread only needs to wait until its worker threads are done, the Wait/Pulse approach is better
  - Let the .NET runtime handle it!

Locating your threads on the Xbox 360

```
Thread.CurrentThread.SetProcessorAffinity
  (new int[] {index});
```

- Set thread affinity ***within*** the worker thread immediately after starting it
  - Don't forget to call it, or your worker thread will be running on the same hardware thread as your main thread
- Only available on Xbox 360 XNA

Check to see if you're on an Xbox 360

```
#if XBOX360
  Thread.CurrentThread.SetProcessorAffinity
    (new int[] {index});
#endif
```

- No way I know of in C# to manually set processor affinity in Windows like on the Xbox 360
- Windows decides what threads run where

Xbox 360 hardware threads

Ind	CPU	Core	Comment
0	1	1	Not available in XNA
1	1	2	Available; main thread; game runs here by default
2	2	1	Not available in XNA
3	2	2	Available; parts of the Guide and Dashboard live here
4	3	1	Available; Xbox Live Marketplace downloads
5	3	2	Available; parts of the Guide and Dashboard live here

Table from Joseph Hall, “XNA Game Studio Express,” p. 608

Advice

- More than one thread per core isn't bad…
- …but more than one processor-intensive task per core is!
- Put most intensive tasks on separate cores, and some less-demanding tasks on those same cores (threads that work in short bursts, disk I/O, etc.)

Advice from Joseph Hall, "XNA Game Studio Express," p. 610

More advice

- Limit number of synchronization points
- Don't lock resources longer than necessary
- Avoid sharing data when possible
- Profile your code before and after to make sure you're getting the performance benefits you expect
  - Very easy to write multithreaded code that performs worse than single threaded!

Advice from Joseph Hall, "XNA Game Studio Express," p. 611