ECE4893A/CS4803MPG: Multicore and GPU Programming for Video Games

Lecture 6: Introduction to XNA, Game Loops, and C# Gotchas

Prof. Aaron Lanterman

School of Electrical and Computer Engineering

Georgia Institute of Technology

Georgia Institute of Technology

Part 1:
Introduction to XNA

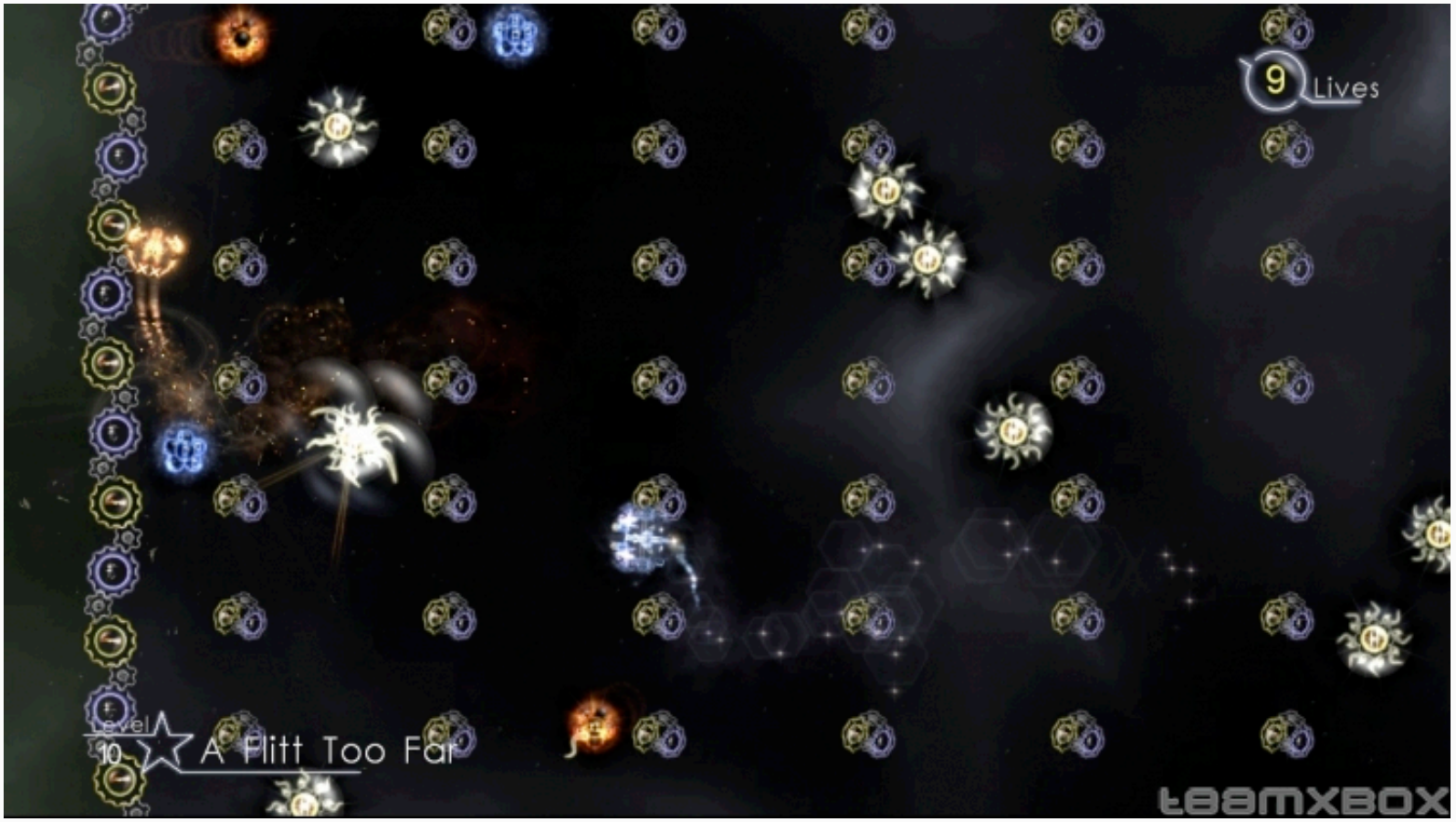
Dungeon Quest

- Developed in 4 days at the 2007 GDC at the XNA contest
- By Benjamin Nitschke and Christoph Rienaecker

Screenshot from
exdream.no-ip.info/blog/2007/07/31/DungeonQuestUpdatedWithSourceCodeNow.aspx


Goblin Wizard
Health
Next Level 3
Points: 175
Time: 01:30
Skills (0 left)
Defence (2)
Speed (0)
Attack (0)

Torpex's "Schizoid" (on Xbox Live Arcade)



Screenshot from http://screenshots.teamxbox.com/screen/68599/Schizoid/

http://www.gametrailers.com/player/28542.html

Georgia Institute of Technology

XNA GS Framework

- Built on Microsoft's .NET
  - Makes MS comfortable with letting "ordinary folks" program on the Xbox 360
- C# is standard language for XNA development
  - But in theory could use Managed C++, VB.NET, etc. on the PC
- Xbox 360 uses .NET Compact Framework
  - Some stuff available in .NET on the PC is missing!
  - Garbage collector on 360 isn't as smart as on the PC
  - Caused the Schizoid team some trouble, as well as one semester of CS4455

Is managed code too slow for games?

- Vertigo Software ported Quake II to Managed C++, got 85% performance of the original C code
  - Should expect to do better if you have the .NET Common Language Runtime in mind from the beginning
- Xbox 360
  - GPU: 337 million transistors
  - CPU: 165 million transistors

XNA GS graphics

- XNA is built on top of DirectX 9
  - Not built on MDX or Managed DirectX
- DirectX 9 has a fixed function pipeline, but XNA doesn't!
  - Everything done with shaders
  - There is a BasicEffect to get you started

From XNA Team Blog, "What is the XNA Framework,"
blogs.msdn.com/xna/archive/2006/08/25/724607.aspx

Why no fixed-function pipeline? (1)

In Microsoft's own words (paraphrased):

- Programmable pipeline is the future
  - Neither Direct3D 10 or Xbox 360 have fixed-function pipeline
- Early adopters and customers said cross-platform goal more important than fixed-function pipeline

From XNA Team Blog, "What is the XNA Framework," blogs.msdn.com/xna/archive/2006/08/25/724607.aspx

Why no fixed-function pipeline? (2)

In Microsoft's own words (paraphrased):

- Fear is someone would start and finish their game using the fixed-function APIs, and then get dozens of errors when they tried to compile it on the Xbox 360
- Better to know your code works on both right from the beginning

From XNA Team Blog, "What is the XNA Framework," blogs.msdn.com/xna/archive/2006/08/25/724607.aspx

Some convenient things about XNA

- Don't need to mess with Win32-ish boilerplate (opening a window, etc.)
- Easy interfacing with the Xbox 360 controller (for both Windows and Xbox 360)
- Storage ("saved games") unified between Windows and Xbox 360
  - On Xbox 360, have to associate data with a user profile, put on hard drive or memory card, etc.
  - XNA "emulates" this on Windows

From XNA Team Blog, "What is the XNA Framework," blogs.msdn.com/xna/archive/2006/08/25/724607.aspx

Hello Bluescreen

From XNA Team Blog, "What is the XNA Framework," blogs.msdn.com/xna/archive/2006/08/25/724607.aspx

```
public class SampleGame : Game {
    private GraphicsComponent graphics;

    public SampleGame() {
        this.graphics = new GraphicsComponent();
        this.GameComponents.Add(graphics);
    }

    protected override void Update() { }

    protected override void Draw() {
        this.graphics.GraphicsDevice.Clear(Color.Blue);
        this.graphics.GraphicsDevice.Present();
    }

    static void Main(string[] args) {
        using (SampleGame game = new SampleGame()) {
            game.Run();
        }
    }
}
```

Careful if you're on Windows x64

- XNA normally targets "AnyCPU"
- Will break when you try to run on x64 machines, since x64 versions XNA framework dlls don't exist (yet – maybe never?)
- Workaround: Change target to x86

Caveats about Xbox 360 development

- Many TVs cutoff 5-10% of the pixels around the edge
  - Keep text & important info away from there
- Xbox 360 handles post processing and render targets a little differently than the PC

Info from Alistair Wallis, “Microsoft XNA: A Primer,” interview with Benjamin Nitschke
www.gamecareerguide.com/features/328/microsofts_xna_a_.php?page=4

Dream Build Play contest

- See http://www.dreambuildplay.com
- This year's contest already over…
- …but keep on the lookout for the 2010 Dream Build Play contest!

XNA Community Games

- See http://creators.xna.com
- Join the XNA Creator's Club
  - The XNA CC memberships students get free from DreamSpark will let you run games on the 360, but may not let you take part in Community Games
- Upload your game, rate content (violence, etc.)
- Peer review – confirm content ratings, check quality
- Can sell your game to Xbox 360 users!

Example: A Fading Melody

XNA CG sales (March 31, 2009)

Find Teddy	1-Feb-09	2,181	70	3.2%	400	$338.80	$237.16
Remote Masseuse	11-Feb-09	55,000	3,500	6.4%	200	$8,470.00	$5,929.00
Exhaust	14-Feb-09	27,256	990	3.6%	400	$4,791.60	$3,354.12
Trajectory	20-Feb-09	1,928	59	3.1%	200	$142.78	$99.95
Tomato Blaster	22-Feb-09	704	25	3.6%	400	$121.00	$84.70
ZoomaRoom	25-Feb-09	4,703	398	8.5%	200	$963.16	$674.21
Alchemist	27-Feb-09	1,356	101	7.4%	200	$244.42	$171.09
ZP2K9	28-Feb-09	19,628	3,386	17.3%	200	$8,194.12	$5,735.88
Snake360 Lite	10-Mar-09	4,798	376	7.8%	200	$909.92	$636.94
Solar	20-Mar-09	8,000	1,466	18.3%	200	$3,547.72	$2,483.40
Clock 24-7	21-Mar-09	4,865	249	5.1%	200	$602.58	$421.81
Totals		350,433	25,049	9.2%		$69,550.80	$48,685.56

From from http://www.gamasutra.com/php-bin/news_index.php?story=22970

Part 2:
Game Loops

Credit to where it is due

- Koen Witters
  - Thinking about game loops
- Shawn Hargreaves
  - Details about XNA's game loop
- Side note: next few slides on game loops contain rough *pseudocode*

Simplest game loop (1)

```
running = true;

while(running) {
    update();
    draw();
}
```

- Draw() has things like `bad_guy.x += 1;`
- What could possibly go wrong?

http://dewitters.koonsolo.com/gameloop.html

Simplest game loop (2)

- Game runs faster on faster hardware, slower on slower hardware
- Less of a problem if hardware is well-defined; Apple II+, Commodore 64, game console
- Try an original Mac game on a Mac II: too fast!
- Big problem on PCs/Macs with varying speed
- Can still be a problem if update time varies from iteration to iteration (i.e. varying number of bad guys)
  - See Defender and Robotron: 2084

FPS dependent on constant GS (1)

```
running = true;
seconds_per_frame = 1/60;

while(running) {
   update();
   draw();
   if (seconds_per_frame_not_elapsed_yet)
       wait(remaining_time);
   else {
      oooops! We are running behind!
   }
}
```

- What could possibly go wrong?

http://dewitters.koonsolo.com/gameloop.html

FPS dependent on constant GS (2)

- Slow hardware:
  - –If fast enough to keep up with FPS no problem
  - –If not: game will run slower
  - –Worst case: some times runs normally, sometimes slower – can make unplayable

http://dewitters.koonsolo.com/gameloop.html

FPS dependent on constant GS (3)

- Fast hardware:
  - Wasting cycles on desktops - higher FPS gives smoother experience, why not give that to the user?
  - Maybe not so bad philosophy on mobile devices – save battery life!
  - Also may not be bad if user is wants to run other processes

http://dewitters.koonsolo.com/gameloop.html

GS dependent on variable FPS (1)

```
running = true;
seconds_per_frame = 1/60;

while(running) {
   update(time_elapsed);
   draw();
}
```

- Use time_elapsed in your state update computations:

```
bad_guy.x += time_elapsed * bad_guy.velocity_x;
```

- What could possibly go wrong?

http://dewitters.koonsolo.com/gameloop.html

GS dependent on variable FPS (2)

- Slow hardware:
  - Game sometimes bogs down, i.e. when lots of stuff is on the screen
    - Slows down player and AI reaction time
  - If time step is too big:
    - Physics simulations may become unstable
    - “Tunneling” (need “swept collision detection”)

http://dewitters.koonsolo.com/gameloop.html

GS dependent on variable FPS (3)

- Fast hardware:
  - –Shouldn't be a problem, right?
  - –What could possibly go wrong?

http://dewitters.koonsolo.com/gameloop.html

GS dependent on variable FPS (4)

- Fast hardware:
  - More calculations per second for some quantity, more round off errors can accumulate
  - Multiplayer game: players with systems with different speeds will have game states drifting apart
  - Good example:
    - www.nuclex.org/articles/xna-game-loop-basics

http://dewitters.koonsolo.com/gameloop.html

Constant GS with max FPS (1)

```
running = true;
seconds_per_gametick = 1/50;
max_gameticks_skipped = 10;
next_gametick_time = current_time();

while (running) {
    loop = 0;
    while (current_time() > next_gametick_time
                    && loop < max_gameticks_skip ) {
        update();
        loop++;
        next_gametick_time += second_per_gametick;
    }
    draw();
}
```

http://dewitters.koonsolo.com/gameloop.html

Constant GS with max FPS (2)

```
running = true;
seconds_per_gametick = 1/50;
max_gameticks_skipped = 10;
next_gametick_time = current_time();

while (running) {
   loop = 0;
   while (current_time() > next_gametick_time
                  && loops < max_gameticks_skip ) {
      update();
      loop++;
      next_gametick_time += second_per_gametick;
   }
   draw();
}
```

- Game updated 50 times per second, render as fast as possible
- If rendering more than 50 times per second, some frames will be the same

http://dewitters.koonsolo.com/gameloop.html

Limits of constant GS with max FPS

- On slow hardware:
  - May have low FPS, but hopefully game will run at normal speed
  - If FPS drops below gameticks_per_second / maximum_gameticks_skipped (5 in previous example), GS slows down
- On fast hardware:
  - Wasting time redrawing the same scene (or, with better logic, twiddling thumbs)
- Balancing act: want fast update rate, but still be able to run on slow hardware

http://dewitters.koonsolo.com/gameloop.html

Balancing act

- Want fast update rate…
- …but still be able to run on slow hardware

http://dewitters.koonsolo.com/gameloop.html

Constant GS indep. of variable FPS

- Update, at say, 25 times per second
  - Player input, AI, etc.
- Render faster on faster graphics hardware
  - Use interpolation to predict where objects should be
  - Makes it look like full game is running at a high frame rate
- Degrades gracefully on slower hardware

http://dewitters.koonsolo.com/gameloop.html

Tasks with different granularity

- Run often:
  - Physics engine location & orientation updates
  - 3-D character display
- Run less often:
  - Collision detection
  - Player input
  - Head-up display
- Run even less often:
  - “immediate A.I.”, networking
- Careful: A.I. might be unstable with larger time steps – not just physics!

Example: MotoGP

- Main game logic: 60 updates per second
  - "input, sound, user interface logic, camera movement, rider animations, AI, and graphical effects"
- Physics: 120 updates per second
- Networking: 4 to 30 updates per second, depending on number of players – more players results in less often updates to conserve bandwidth

http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx

XNA game loop: fixed step

- Game.IsFixedTimeStep = true; (default)
- XNA calls Update() every “TargetElapsedTime” (defaults to 1/60 seconds)
  - Repeat call as many times as needed to catch up with current frame (in XNA >= 2.0)
- XNA hopefully calls Draw(), then waits for next update
- If Update+Draw time < TargetElapsedTime, we get
  - Update
  - Draw
  - Hang out for rest of time (nice on Windows so other processes can run)

http://blogs.msdn.com/shawnhar/archive/2007/11/23/game-timing-in-xna-game-studio-2-0.aspx

XNA may get behind (1)

- Why would

  Update+Draw time > TargetElapsedTime?
  - Computer slightly too slow
  - Computer way too slow
  - Computer mostly fast enough, but may have too much stuff on screen, big texture load garbage collection
  - Paused program in debugger

http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx

XNA may get behind (2)

- What happens if Update+Draw time > TargetElapsedTime?
  - Set GameTime.IsRunningSlowly = true;
  - Keep calling Update (without Draw) until caught up
    - Makes sure game is in right state with Draw finally happens
  - If too far behind… punt

http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx

When XNA gets behind (1)

- If computer slightly too slow: If can't handle Update+Draw in one frame, can probably handle Update+Update+Draw in two frames
  - May look jerky but should play OK
- If computer way too slow (i.e. Update alone doesn't fit in a single frame): we are doomed
- In both above cases, a clever program could see that GameTime.IsRunningSlowly == true and reduce level of detail
  - Most games don't bother

http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx

When XNA gets behind (2)

- If particular frame took too long: call update extra times to catch up, then continue as normal
  - Player may notice slight glitch
- If paused in debugger: XNA will get way behind and give up, but will continue running OK when debugger resumed

http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx

"Heisenberg Uncertainty Principle"

- If you put in breakpoints, may notice Update being called more often than Draw, since the breakpoint makes you late
- Examining the timing of a system changes the timing!

http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx

XNA game loop: Variable Step

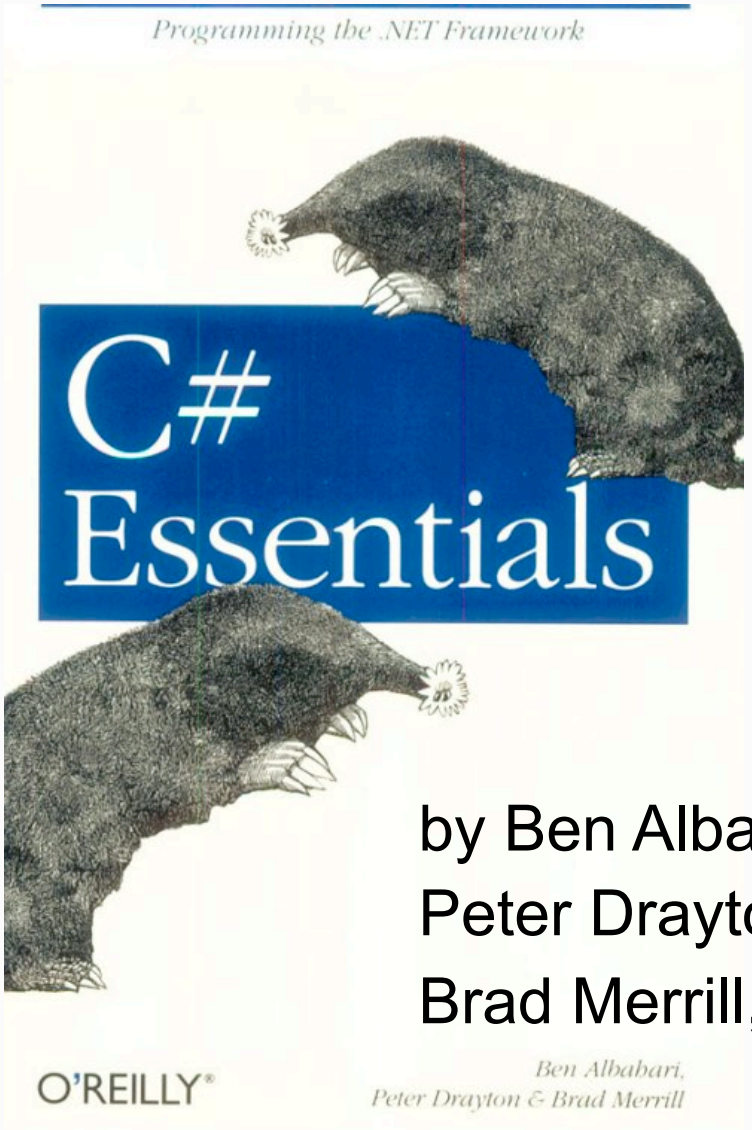
- Game.IsFixedTimeStep = false;
  - Update
  - Draw
  - Repeat
  - (more or less)

- Update should use elapsed time information

http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx

Part 3:
C# Gotchas

References



Great article:

Jesse Liberty, "Top ten traps in C# for C++ programmers"

www.ondotnet.com/pub/a/oreilly/dotnet/news/programmingCsharp_0801.html

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

Georgia Institute of Technology

Value vs. reference types

- Like C++, C# has user defined types
- C# also makes a distinction between value types and reference types
- Value types:
  - Intrinsic types and structs
  - “Passed by value” (copied)
  - Stored on the stack (unless part of a reference type)
- Reference types:
  - Classes and interfaces, and “boxed” value types
  - “passed by reference” (implicit pointer)
  - Variables sit on the stack, but hold a pointer to an address on the heap; real object lives on heap

Boxing and unboxing

- Boxing allows value types to be treated as reference types
  - Value boxed inside an object
  - Unboxed to get original value back
- Everything in C# is derived from “Object,” so everything can be implicitly cast to an object
- Unboxing must be done explicitly

Boxing and unboxing example

```
using System;
public class UnboxingTest
{
    public static void Main()
    {
        int i = 123;

        //Boxing
        object o = i;

        // unboxing (must be explicit)
        int j = (int) o;
        Console.WriteLine("j: {0}", j);
    }
}
```

If o is null or not an `int` an InvalidCastException is thrown

From Jesse Liberty, “Top ten traps in C# for C++ programmers,” www.ondotnet.com/pub/a/oreilly/dotnet/news/programmingCsharp_0801.html

Structs vs. classes

- Structs are value types
  - More efficient when used in arrays
  - Less efficient when used in collections
    - Collections expect reference types, so structs must be “boxed” - boxing has overhead
  - Support properties, methods, fields, and operators…
  - …but not inheritance or destructions
- Classes are reference types
  - May be more efficient when used in collections

Reference parameters

- C, C++, and C# allow a function to only return one value
- In C++ and C#, you can get around this by passing in pointers
- In C#:
  - Reference types in the parameter list may be changed by the function
  - To let a function change a value type in the parameter list, can use an explicit `ref` keyword:

`ref` must be used in both declaration and call

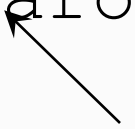
```
public void Changer(ref int x)

Aaron.Changer(ref int aaronx);
```

Variables must be initialized

```
public void Changer(ref int x)

int aaronx;
Aaron.Changer(ref int aaronx);
```



C# will give a compile-time error since `aaronx` has not been initialized

In general, variables in C# must be assigned before being passed into a function

A clunky workaround

```
public void Changer(ref int x)

int aaronx = 0;
Aaron.Changer(ref int aaronx);
```

The `out` keyword

```
public void Changer(out int x)


int aaronx;
Aaron.Changer(out int aaronx);
```


`out` keyword like **`ref`**, except it tells C# that it's OK for the value to be undefined

C# will demand that you assign `aaronx` before the function returns!

C# Finalizers (1)

```
~MyClass()
{
  // your code to release unmanaged resources
  // used by object
}
```

is syntactic sugar for

```
MyClass.Finalize()
{
  // your code to release unmanaged
  // resources used by object
  base.Finalize();
}
```

Your finalizer should not try to deal with other C# reference objects - only deal with unmanaged resources!

Adapted from Jesse Liberty, "Top ten traps in C# for C++ programmers," www.ondotnet.com/pub/a/oreilly/dotnet/news/programmingCsharp_0801.html

C# Finalizers (2)

- Finalizer will be called when the .NET garbage collector decides to call it
  - You don't get to decide when it's called
- Only define a finalizer if you really need one
  - Calling it involves some overhead

Pop quiz: C

- What is the value of `b` after this code is run (assume C code)?

```
a = 7;
b = 3;
if (a = 5)
{
    b = 10;
}
```

Booleans in C#

- In C, `0` is false, “anything else” is true
- In C#, this code will give a compile time error
  - C# has distinct Boolean values, `true` and `false`

```
a = 7;
b = 3;
if (a = 5)
{
    b = 10;
}
```

C# arrays are objects

Java: `int arr1[];`

C#: `int[]arr1;`

```
arr1 = new int[5];
arr1 = new int[5]{10,20,30,40,50};
int[] arr2 = new int[5] {10,20,30,40,50};
int[] arr2 = {10,20,30,40,50};
```

Multi-dimensional arrays

```
string[,] bingo;

bingo = new string[3,2]  {{"A","B"},
  {"C","D"},{"E","F"}};
bingo = new string[,] {{"A","B"},
  {"C","D"},{"E","F"}};

string[,] bingo = {{"A","B"},{"C","D"},
  {"E","F"}};
```

Jagged arrays

- Arrays of arrays

```
int[][] arr =
new int[][]
   {new int[] {10,11,12}, new int[] {13, 14,
   15, 16, 17}};
```

Array iteration

```
int[] arr = {16, 17, 18};
foreach (int x in arr)
{
 System.Console.WriteLine(x.ToString());
}
```