**Dungeon Quest**

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

[Image of Dungeon Quest]

**Torpex’s “Schizoid” (on Xbox Live Arcade)**

[Image of Torpex’s “Schizoid”]

[Link to Schizoid](http://www.gametrailers.com/player/28542.html)
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

Why no fixed-function pipeline?

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
- 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

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


Hello Bluescreen

```csharp
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();
        }
    }
}
```


XNA strengths & weaknesses: audio

• Uses XACT, Microsoft’s Audio Creation Tool
• Nice for modifying sound effects directly in XACT until you like them
• No support for .mp3 or .wma
• Xbox 360 has XMA, and Windows has ADPCM, but game still 3-5 bigger than it might otherwise be

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

Benjamin Nitschke’s Rocket Commander

Original 10M, XNA version 50M due to larger audio files
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)
- 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

Dream Build Play contest

- See http://www.dreambuildplay.com
- This year’s entries are due Sept. 23
  - Too late for this year, probably…
  - But keep on the lookout for the 2009 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!
Introduction to C# - emphasis on “gotchas”  
Great article:  
Jesse Liberty, “Top ten traps in C# for C++ programmers”  
by Ben Albahari,  
Peter Drayton, and  
Brad Merrill, 2001

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  
using System;  
public class UnboxingTest  
{  
  public static void Main()  
  {  
    int i = 123;  
    object o = i;  
    // Boxing  
    // unboxing (must be explicit)  
    int j = (int) o;  
    Console.WriteLine("j: {0}", j);  
  }  
}

From Jesse Liberty, “Top ten traps in C# for C++ programmers.”  
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 destructors
- 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:

```csharp
public void Changer(ref int x)

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

Variables must be initialized

```csharp
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

```csharp
public void Changer(ref int x)

int aaronx = 0;
Aaron.Changer(ref int aaronx);
```
The **out** keyword

```csharp
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

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

is syntactic sugar for

```csharp
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

- 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)?

```c
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

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