

XBox Project Cluster, E2006

Course Introduction

Ken Friis Larsen

kfl@itu.dk

IT University of Copenhagen

Monday September 4, 2006

Today's Subjects

- ▶ General course information
- ▶ Weekly exercises and the exam
- ▶ Course content and motivation
- ▶ Web publishing and Web applications
- ▶ Detailed course content
- ▶ HTML in 21 minutes

What Is XNA

- ▶ The XNA Framework, a set of managed code development libraries for creating games for Windows and the XBox 360.
- ▶ The XNA Framework Content Pipeline, a set of tools that allow developers to more incorporate 3D content into their games.
- ▶ XNA Game Studio Express. Contains, among other things, some documentation, how-tos, and starter kits that demonstrate how to use the content pipeline and XNA Framework.
- ▶ XNA's Not Acronymed

Getting Started

- ▶ Download and install Visual C# Express
- ▶ Make sure you have the latest DirectX runtime
- ▶ Download and install XNA Game Studio Express
- ▶ Browse the the “Getting Started” content from within Game Studio Express.

Why Build Your Own Game Engine

- ▶ Differentiate yourself
- ▶ Control
- ▶ Non-mainstream requirements

Managed Framework For Games???

- ▶ Actually Managed Runtimes in the game world are common, but usually they are made in-house.
- ▶ With the current technology you can get quite decent performance.
- ▶ Garbage Collection is the big advantage

What is C#

- ▶ A class-based single-inheritance object-oriented programming language with a managed execution environment.
- ▶ Similar to Java, but with many improvements and complications.
- ▶ Designed to be the main programming language for Microsoft .NET, aka Common Language Infrastructure (CLI).
- ▶ The syntax of C# looks a lot like Java:

```
using System;
class Program {
    static void Main(string[] args) {
        Console.WriteLine("Hello World!");
    }
}
```

Comparison of C# and Java

Most of C# is immediately recognizable to a Java programmer. Some differences:

- ▶ Virtual and non-virtual instance methods.
- ▶ Implicit boxing and unboxing of primitive types (also in Java 5.0).
- ▶ Properties—field-style method calls.
- ▶ Indexers—array-style method calls.
- ▶ Enumerators (iterators) and the foreach statement (also in Java 5.0).
- ▶ Operator overloading—as in C++.
- ▶ Delegates—method closures.
- ▶ Value types and structs—allocated on stack, inlined in objects and arrays, copied on assignment and so on.
- ▶ Enum types—as in C/C++ (also in Java 5.0).
- ▶ Reference parameters (ref and out)—much like Pascal, Ada, C++.
- ▶ Variable-arity methods (params modifier; also in Java 5.0).
- ▶ No inner classes, no throws clause on methods.
- ▶ Unsafe code with pointer arithmetic and so on—discouraged, but possible.

Garbage Collection

- ▶ Amortized cost of allocations in this model can be excellent.
- ▶ Long term fragmentation is less of a problem
- ▶ Locality can be good due to compaction
- ▶ Easy to use model, immune to classic “leaks” and wild pointers

Garbage Collection Models

- ▶ In the .NET framework on the PC has a **Generational GC**
- ▶ he Compact .NET framwork uses an **compacting mark and sweep collector**

Generational GC

- ▶ The heap is divided into generations
- ▶ New objects are allocated in generation 0
- ▶ We collect generation-wise
- ▶ If an object survives a collection in generation n it is moved to generation $n + 1$
- ▶ Objects within a generation are all roughly the same age.

Compacting Mark and Sweep GC

- ▶ Two phases:
 - ▶ Live objects are marked
 - ▶ The whole heap is swiped, compacting the live objects.
- ▶ The compacting is good for cache performance

Good Things To Know About GC

- ▶ Allocations are cheap
- ▶ Objects that are allocated close together in time tend to be close together in space
- ▶ Full collections can be very expensive because the heap could be arbitrarily big
- ▶ Only live objects are interesting
- ▶ Leaks with a GC is usually when you keep objects alive for too long

Game Engine Design

- ▶ Scene graph
- ▶ Collision detection
- ▶ ...