Today's Lecture

- Development – Windows
  - Project creation
  - Project debugging
- C# Primer
  - Variables / classes
  - Console I / O
  - STL Equivalents
  - Properties
  - Delegates

Reminders
- Wiki Editing
- Blog Creation
- C# Programming

Interface Highlight – CSE - 2007

UPCOMING EVENTS

Department Calendar
Small Group Exercise

- Did the website get better or worse?
- Are there any aspects that you like / dislike when visiting websites?

Split into groups of 2-4 students and answer the two above questions.
System Interface Lab

□ Fitzpatrick 355S
  ■ Access via passcode
  ■ 3x HP Multi-Touch (Windows Vista)
    □ Visual Studio, Microsoft Surface SDK, XNA Game Studio
  ■ 1x Nintendo Wii
    □ 4x Wiimotes/charger, 1x Balance Board
  ■ 2x Microsoft Surface
    □ 1 unit – Commercial -> Main Office
    □ 1 unit – Developer -> Stay in lab

Lab Demo – Friday - Retrospective

What were your impressions of the Surface?

C# Programming

□ Buzzwords
  ■ General purpose
  ■ Type-safe
  ■ Object-oriented
□ Other details
  ■ Chief architect – Anders Hejlsberg
    □ Turbo Pascal
    □ Delphi
  ■ Platform neutral
    □ Technically (if you ignore .NET)
Object-Oriented

Key aspects
- Classes / objects
- Encapsulation
  - Public vs. private
- Inheritance
  - Protected
- Polymorphism
  - Diversity in child classes
  - List holds many with common base type

Distinctive Features

Unified Type System
- All data shares the same type
- Including basic ones
  - char, int, bool
  - Map out to larger class
- Every object has a ToString method

Classes and interfaces
- Definition of a type
  - Cannot instantiate it
- Can only derive from a single class in C#

Distinctive Features (cont)

Properties, methods, events
- Methods
  - Normal type of functions
- Properties
  - Can replace get / set
- Events
  - Use to trigger
  - Gather together -> delegates
### Other Aspects

- **Type-safe**
  - Can only interact through defined protocols (functions)
  - Static typing
    - Compile-time checking
  - Dynamic typing
    - Run-time checking via .NET
    - CLR – Common Language Runtime
- **Strongly typed**
- **Garbage collection**
  - No pointers

...Isn't this just Java?

### CLR, WPF, XNA, WTF?

- **C#**
  - Base language – very similar to Java
- **CLR / .NET**
  - Makes C# go on Windows
- **WPF**
  - Windows Presentation Foundation
  - How we will write out GUIs
- **XNA**
  - Successor for DirectX
  - Graphics focus -> think games

We will not be doing Windows Forms

### What will we cover?

- **Each class**
  - Neat C# / WPF trick
- **Occasionally**
  - Broader dive into C# / WPF
- **Today**
  - Hello, World
    - Create / compile code
  - Course List
    - Properties
Create a Project

C# Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

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

Similar to `#include`

Adding a namespace

Namespace for this code

Class name -> Parent, Interfaces

```
static
```

Non-instantiation of class

Namespaces

When you type something, it explores the currently added namespaces to help you out

```
C#:
```

No header / source file demarcations
**Project aka Solution**

Mainly, we will just see `.cs` and `.xaml` (later) files

Automatically generated makefile (it does exist)

Solution can have multiple projects

---

**Hello, World**

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

**Console**

Collection of `static` methods ~ `cout`, `cin` from C++

**Methods**

- `WriteLine`
- `ReadLine`

---

**Building the Solution**

- **F6** – Compile / Link

Code will also auto-compile
Testing the Code

Non-interactive console apps are hard to test

F5 – Compile / Link  F9 – Toggle Breakpoint

File Exploration

Use the Command Prompt

Start->
Accessories->
Command Prompt
(right click to Run as Admin in Vista)
Are we a C# Jedi yet?
Not really
Much, much more to cover
Properties
STL analogues

Simple student list
- Revisit the classic list of stuff
  - Student
    - First name
    - Last name
    - GPA
    - Major
  - Make a list of them and display it
- Homework 2
  - List and manipulate files

Add a new class....
Name != Class
Class Design – C++

class Student
{
private:
    String m_sFirstName;
    // …

public:
    Student ()
    {  …  }
    void getFirstName ();
}

Class Design – C#

□ Several changes
   ▪ No header file
   ▪ Per item ACL
       □ Designate each item / function
       □ Default is private
   ▪ Similar modes
       □ public, protected, private
   ▪ Can also make “global”
       □ Static – do not need to instantiate

Big Change - Properties

□ Normally do get / set functions
   ▪ getFirstName()
   ▪ setFirstName()

class Student
{
    String m_sFirstname;

public String FirstName
{
    get { return m_sFirstName; }
    set { m_sFirstName = value; }
}
Let’s see that again....

```csharp
public Type PropertyName
{
    get { // Stuff to get }
    set { // Stuff to set }
}
```

- **get**: Return type must match property type
- **set**: Value is the same as the property type

Using Properties

```csharp
Student theStudent;
theStudent = new Student();

// Old School
theStudent.setFirstName("Aaron");
Console.WriteLine(theStudent.getFirstName());

// New school
theStudent.FirstName = "Aaron";
Console.WriteLine(theStudent.FirstName);
```

Everything is a "pointer" (except primitives*)

Automatic Properties

```csharp
public String FirstName
{
    get;
    set;
}
```

- **New feature in .NET 3.0**: Automatically creates the variable underneath
- **Does it range check?**
Where do we use this?

Real power is when we tie it to controls

Student.FirstName -> TextField.DataBindings

Any changes in the control automatically update our variable

WHOOOT!

Populate the class

- See example code on wiki

Override a function

```java
public override String ToString()
{
    String sTemp = FirstName + " " + LastName + " " + GPA + " " + Major + " " + Major;
    return sTemp;
}

override -> virtual
Part in child
```

toString exists in every class
Part of base object
**Code Example**

```csharp
Student theStudent;
theStudent = new Student();
theStudent.firstName = "OOhm";
theStudent.lastName = "Doe";
theStudent.GPA = (float) 3.67;
theStudent.major = "Cpse E";
```

---

**Collections**

- What is similar to STL Vector?
  - List

```csharp
List<Type> varName;
varname = new List<Type>();
```

```csharp
List<Student> listStudents;
listStudents = new List<Student>();
```

---

**Using the List type**

```csharp
This list = new List<Student>();
Student theStudent;
theStudent = new Student();
theStudent.firstName = "John";
theStudent.lastName = "Doe";
theStudent.GPA = (float) 3.67;
theStudent.major = "ECE";
theList.Add(theStudent);
theStudent = new Student();
theStudent.firstName = "Jane";
theStudent.lastName = "Doe";
theStudent.GPA = (float) 3.70;
theStudent.major = "ECE";
theList.Add(theStudent);
```

- Add -> push_back
- Access like an array
If we want to have it iterate inside the list, we have to make our own holder class that inherits or has an internal List class.

Questions?

- Homework 2
- Hw1a: Wiki Editing
- Hw1b: Blog Creation
- Week 2 - Blog