Csharp Coding Guidelines

From Unify Community Wiki

Contents

Introduction

This guide is by no means a bible on how to write good code, but shows how to format code to make readability easier, and code cleaner.

Style Guidelines

Below are good practices for code styling

Bracing

Open braces should always be at the beginning of the line after the statement that begins the block. Contents of the brace should be indented by 1 tab or 4 spaces. For example:

```
if (someExpression)
{
    DoSomething();
}
else
{
    DoSomethingElse();
}
```

“case” statements should be indented from the switch statement like this:

```
switch (someExpression)
{
```

```
    case 0:
       DoSomething();
       break;

    case 1:
       DoSomethingElse();
       break;

    case 2:
       {
          int n = 1;
          DoAnotherThing(n);
       }
       break;
}
```

Braces should never be considered optional. Even for single statement blocks, you should always use braces. This increases code readability and maintainability.

```
for (int i=0; i<100; i++) { DoSomething(i); }
```

Single line statements

Single line statements can have braces that begin and end on the same line.

```
public class Foo
{
   int bar;

   public int Bar
   {
      get { return bar; }
      set { bar = value; }
   }

}
```

It is suggested that all control structures (if, while, for, etc.) use braces, but it is not required.

Commenting

Comments should be used to describe intention, algorithmic overview, and/or logical flow. It would be ideal, if from reading the comments alone, someone other than the author could understand a function's intended behavior and general operation. While there are no minimum comment requirements and certainly some very small routines

need no commenting at all, it is hoped that most routines will have comments reflecting the programmer's intent and approach.

Comment Style

The // (two slashes) style of comment tags should be used in most situations. Where ever possible, place comments above the code instead of beside it. Here are some examples:

```
// This is required for Controller access for hit detection
FPSController controller = hit.GetComponent<FPSController>();


// Creare a new ray against the ground
//
Ray ray = new Ray(hit.transform.position, -Vector3.up);
```

Comments can be placed at the end of a line when space allows:

```
public class SomethingUseful
{
    private int           itemHash;             // instance member
    private static bool   hasDoneSomething;     // static member
}
```

Spacing

Spaces improve readability by decreasing code density. Here are some guidelines for the use of space characters within code:

Do use a single space after a comma between function arguments.

Right:

```
Console.In.Read(myChar, 0, 1);
```

Wrong:

```
 Console.In.Read(myChar,0,1);
```

Do not use a space after the parenthesis and function arguments

Right:

```
CreateFoo(myChar, 0, 1)
```

Wrong:

```
 CreateFoo( myChar, 0, 1 )
```

Do not use spaces between a function name and parenthesis.

Right:

```
CreateFoo()
```

Wrong:

```
CreateFoo ()
```

Do not use spaces inside brackets.

Right:

```
x = dataArray[index];
```

Wrong:

```
 x = dataArray[ index ];
```

Do use a single space before flow control statements

Right:

```
while (x == y)
```

Wrong:

```
while(x==y)
```

Do use a single space before and after comparison operators

Right:

```
if (x == y)
```

Wrong:

```
if (x==y)
```

Naming

- **Do not** use Hungarian notation
- **Do not** use a prefix for member variables (_, m_, s_, etc.). If you want to distinguish between local and member variables you should use “this.” in C# and “Me.” in VB.NET.
- **Do** use camelCasing for member variables
- **Do** use camelCasing for parameters
- **Do** use camelCasing for local variables
- **Do** use PascalCasing for function, property, event, and class names
- **Do** prefix interfaces names with “I”

- **Do** not prefix enums, classes, or delegates with any letter

File Organization

- Source files should contain only one public type, although multiple internal classes are allowed
- Source files should be given the name of the public class in the file
- Classes member should be alphabetized, and grouped into sections (Fields, Constructors, Properties, Events, Methods, Private interface implementations, Nested types)

Example

```
using System;
using UnityEngine;

public class MyClass : MonoBehavior
{
    // fields
    int foo;

    // properties
    public int Foo { get { … } set { … } }

    // methods
    void MyMethod(int number)
    {
        int value = number + 2;
        Debug.Log(value);
    }
}
```

More Soon

Retrieved from "http://wiki.unity3d.com/index.php?title=Csharp_Coding_Guidelines&oldid=17000"
