C#编程规范 v0.2

更新时间：2018-8-28

1 介绍

文档描述本公司C#语言编程规范。

2 源代码文件

2.1 文件名

文件名跟顶层类名相同，后缀.cs。

2.2 编码

使用UTF-8编码。

2.3 结构

源代码文件按顺序由以下组成：

1. using语句
2. namespace
3. 类定义

以上三种代码之间必须加一空行，即最后一个using和namespace之间加空行，namespace和类定义之间加空行。

2.3.2 using语句

using按如下顺序分组：

1. C#标准库
2. Unity3D的库
3. 本项目中其他namespace
4. 第三方库

只using所需要的类，不需要的清除掉其using语句。

2.3.3 namespace

顶层命名空间必须是项目名称或Joywinds，子命名空间可以根据模块功能来定。

2.3.4 类定义

一个源代码文件只包含一个顶层类，顶层类里可以包含子类。

2.3.4 示例

```
using System;
using System.Collections;
using UnityEngine;
using Joywinds;
using Joywinds.Helper;
using FastJson;

namespace Joywinds {

public class UI : MonoBehaviour {

    void Start() {
        Log.Info("Hello World");
    }

}

}
```

3 格式

3.1 大括号

左大括号前不换行，左大括号后换行。右大括号前换行，右大括号后不换行除非是整个代码块最后一个右大括号。

示例:

```
public void Method() { // 左大括号前不换行，左大括号后换行
    if (something) {
        DoSomething();
    } else { // 右大括号前换行，右大括号后不换行因为后面还有else
        NotDoSomething();
    } // 右大括号后换行因为是整个if/else代码块最后一个右大括号
    Foobar();
}
```

如果代码块为空或只有一行，左右大括号可以在同一行。例如：

```
public virtual void Method() {}
public bool IsOk() { return true; }
```

3.2 缩进

使用四个空格来缩进，不使用Tab。设置你的编辑器将Tab转换为四个空格。

3.2.1 顶层类不缩进

例如：

```
namespace Joywinds {

public class Foo {

    void Bar() {}

}

}
```

3.2.2 内部类，成员变量，成员函数缩进

例如：

```
public class Foo {

    private int m_inited = true;

    public enum Type {
        A,
        B
    };

    public void Bar() {}

}
```

3.3 空行和空格

3.3.1 空行

空行用来对代码行分组，比如两个函数之间，私有变量和公有变量之间或两段不同的逻辑代码。例如:

```
public class Foo {

    private bool m_init = false;
    private GameObject m_parent;

    public int count { get; set; }
    public bool isOk { return m_parent != null; }

    public void Bar() {
        int val = 1;
        Func1(val);

        int val2 = 2;
        Func2(val);
    }

    private void Help() {}

}
```

类定义第一行和最后一个各加一个空行 例如：

```
public class Foo {
                          // <-- 此处加一个空行
    void Bar() {}
                          // <-- 此处加一个空行
}
```

避免使用连续多个空行，尽量使代码紧凑。 例如：

```
public void Foo() {}


puiblic void Bar() {} // Bad, there are two blank line above
```

3.3.2 空格

除语法要求空格外，还应遵循以下原则：

1. if/elseif/for/while/swith等与之后的括号之间加一个空格
2. 函数参数之间加一个空格
3. 函数参数列表与括号之间不加空格
4. 函数默认参数等号前后不加空格
5. 右括号与之后的左大括号加空格

例如：

```
public void Foo(int a, int b, int c=0) {
    ...
}
```

避免使用多余的空格 不好的例子如下：

```
public void Foo( int    a) {}

public void Bar(int b)     {
    int a   = 1;
}
```

3.3 注释

注释使用 `/**/` 和 `//` 都可以，每行注释的首个字母前必须有空格，并且注释必须用**英文**。例如：

```
/*
* This is
* a comment.
*/

// This is a comment
// too.
```

3.4 修饰符

3.4.1 修饰符的排列按以下优先级：

```
public protected private abstract virtual override static final
```

例如：

```
public virtual void Do() {}
protected override void OnInit() {}
public static int Add(int a, int b) { return a + b; }
public static final bool DEBUG = true;
```

3.4.2

`private`修饰符省略不写以减少代码

4 命名

良好的命名增强程序的可读性。命名尽量不要用简写，除非是大部人都知道其意思，例如HUD，UI。命名应该准确表达其含义，避免出现有歧义或者无意义的名字。

4.1 namespace

格式：`UpperCamelCase` 嵌套namespace用点号分开，例如`Joywinds.Data`

4.2 类名/枚举/结构体

格式：`UpperCamelCase` 通常使用名词，例如`class View, enum EventType, struct Vector`，枚举成员格式与此规则相同。

4.3 函数名

格式：`UpperCamelCase` 函数名通常使用动词，例如`Move`， `Enable`

4.4 变量名

4.4.1 私有成员变量名

格式：`m_camelCase` 加一个`m_`前缀，其后是驼峰命名法，首字母小写。

4.4.2 公有成员变量名

格式：`lowerCamelCase` 公有变量必须定义成属性除非是Inspector变量。在内部使用本类的公有变量时尽量带上this，以便与函数参数和临时变量区分开。

例如：

```
public bool isEnabled { get; private set; }

public void Enable() {
    this.isEnable = true;
}
```

4.4.3 函数参数和临时变量

格式：`lowerCamelCase` 参数和临时变量以小写字母开始。

4.4.4 静态私有成员变量名

格式：`s_camelCase` 加一个 `s_` 前缀，，其后是驼峰命名法，首字母小写。

4.4.5 const和readonly变量

格式：`UPPER_CASE_NAME` 所有字母大写，单词之间用下划线分隔。