Coding Guidelines

# Bracing

## Open Braces

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) //另一行再写

{

DoSomething();

}

**else**

{

DoSomethingElse();

}

## Switch Statements

Switch statements should be indented from the switch statement as in the following example:

**switch** (someExpression)

{

**case** 0:

DoSomething();

**break**;

**case** 1:

DoSomethingElse();

**break**;

**case** 2: //case下如果有多个语句必须加{}

{

**int** n = 1;

DoAnotherThing(n);

}

**break**;

}

## Optional Braces

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); } //任何时候都要使用{}

## 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; } //只有一行的语句可以直接在同一行写

set { bar = value; }

}

}

# Comments

## Comment Usage

Comments should be used to describe intention, algorithmic overview, and/or logical flow. A function’s intended behaviour should be understood by reading the comments alone. Most, if not all, routines should have comments describing their intent.

There are no strict requirements on the minimum number of comments in a file.

## Comment Style

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

// This is required for Controller access for hit detection //注释必须写在语句上面

FPSController controller = hit.GetComponent<FPSController>();

// Create a new ray against the ground

## End of Line Comments

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:

## Function Arguments

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

Right:

Console.**In**.Read(myChar, 0, 1); //加空格

Wrong:

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

Do not use a space after the parenthesis and function arguments

Right:

CreateFoo(myChar, 0, 1) //最后面不需要空格

Wrong:

CreateFoo( myChar, 0, 1 )

## Functions

Do not use spaces between a function name and parenthesis.

Right:

CreateFoo() //函数名和（）中间不使用 空格

Wrong:

CreateFoo ()

## Brackets

Do not use spaces inside brackets.

Right:

x = dataArray[index]; //不要在［］使用空格

Wrong:

x = dataArray[ index ];

## Flow Control Statements

Do not use a space before flow control statements

Right:

**while**(x == y) //不要在控制语句前使用空格

Wrong:

**While** (x==y)

## Comparison Operators

Do use a single space before and after comparison operators

Right:

**if**(x == y) //要在＝＝等运算符两边使用空格

Wrong:

**if**(x==y)

# Naming

We should use the following naming conventions:

## Hungarian Notation

Do not use Hungarian notation.//不要使用匈牙利命名法

## Prefixes //使用前缀this，不要使用(\_, m\_, s\_, etc.).

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”.

## Casing // 大小写约定

// camelCasing骆驼命名法：除第一个单词的所有单词的首字母大写

//PascalCasing 第一个单词首字母也要大写

Use camelCasing for member variables, parameters, local variables. Use PascalCasing for function, property, enumeration, event, and class names.

## Prefixes //接口前缀 I

Prefix interface names with “I”. Do not prefix enums, classes, or delegates with any letter. I(interface)

## Constants //常量全部大写

Constants should be in ALL\_CAPS.

# File Organization

## File Contents

Source files should contain only one public type - although multiple internal classes are allowed.

## File Name //文件名必须和class名一样

Source files should be given the name of the public class in the file.

## Class Organisation //分组

Class members should be grouped into the following 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);

}

}

## Regions //不要使用＃region

Do not use regions.

# Enumerations //不要指定枚举的数值

Do not assign values in enumerations. If you need to have values assigned then constant values should be preferred.