Coding Standards

File Formatting

Indentation

Each new level of code signaled by the presence of a { on the previous line should be indented using 3 spaces and decremented by 3 spaces at the } character. Tabs should be avoided.

Libraries

Libraries should be ordered alphabetically at the head of a file. If multiple sub libraries are being used, the top-level library should be at the head of this list

```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManager;
using UnityEngine.UI;
```

Whitespaces

Developers should follow the spacing guidelines below:

· All functions should be separated by at least 2 empty lines in code

· All logic/arithmetic operations done in code should have variables and operators separated by one space

· Place additional blank lines to show separation between unique sections of code

```
public void ClassMethodOne()
{
    numThree = numOne + numTwo;

    Console.WriteLine("NumThree: " + numThree);
}


0 references
public void ClassMethodTwo()
{

}
```

Name Conventions:

Folders

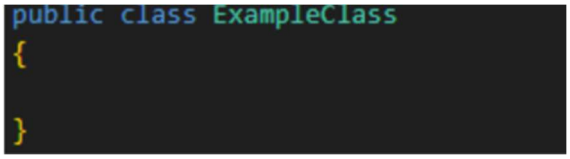
Folders should follow the PascalCasing format, where the first letter of every word is capitalized. No spaces, underscores, numbers, or special characters are used in the name.

Files

File names should always be named like the class they contain. If more than one class is present in a file, the more relevant class should be used to name the file. File names should follow PascalCasing format, where the first letter of every word is capitalized. There should be no spaces, underscores, numbers, or special characters used in the name.

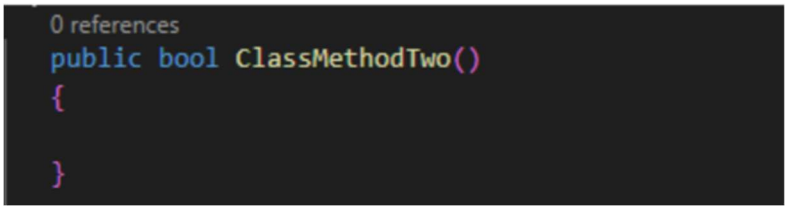
Classes

All class names should follow the PascalCasing format, where the first letter of every word is capitalized. There should be no spaces, underscores, numbers, or special characters used in the name. The name of the class should match the name of the file it resides in.



Functions

All function names should follow the PascalCasing format. There should be no spaces, underscores, numbers, or special characters used in the name. The starting '{' should appear on the line after the function name.



Variables

All variables should follow the camelCasing format, where the first word of the variable is lower case, and all other words capitalized. There should be no spaces, underscores, numbers, or special characters used in the name.

```
public int numOne;
0 references
public int numTwo;
0 references
public int numThree;
```

Documentation

Files

At the beginning of a file before any code or libraries, a comment should be formatted should be written as shown below.

```
/*
* Filename: ExampleClass.cs
* Developer: First Last
* Purpose: This file shows examples for coding standards
*/
```

Classes

Class comments should be put right above the class declaration. As shown below, the first line should contain a summary of the class. Member variables should also be included with a space between the summary and itself as shown below.

Example:

```
/*
* Summary: Example of a correctly formatted class
*
* Member Variables:
* isNumber - a boolian that displays whether or not another variable is a number
*/
0 references
public class ExampleClass
{
    0 references
    bool isNumber = true;
}
```

Functions

Function comments should be put right above the function declaration with the following format. Comments should include a summary, parameters, and returns. If the function returns void, then the returns section is not needed.

Example:

```
/*
* Summary: Checks if the given parameter is 0
*
* Parameters:
* number - the given number that will be checked
*
* Returns:
* Boolian - Returns true if number is 0, returns false if not
*/
0 references
public bool ClassMethodTwo(int num)
{
    if(num == 0)
    {
        return true;
    }

    return false;
}
```

Prefabs

Comment Prefabs Name: Semi-generic Summary: include keywords Description: more detail of prefab, troubleshooting guide, integration instructions, etc.

Other Comments

Any additional comments should be placed above the line of code as shown below in the example. Comments should answer the question likely

```
public bool ClassMethodTwo(int number)
{
    // if statement to check number
    if(number == 0)
    {
        // number is 0
        return true;
    }

    // number is not 0
    return false;
}
```