Unity Engine CSharp Style Guide V4.0.0

hoanglongplanner

2023.10.09

Unity Engine CSharp Style Guide V4.0.0

Author: hoanglongplanner

Organization: ViolettaLappy

Last Updated: 2023.10.09

Table of Contents

Introduction

This just a style guide for development in Unity Engine that aim for project development. NOT FOR 3RD PARTY EXTENSION.

You don't need to uphold yourself to any standard, as long as you can work comfortable and get to the end product at the end.

The cons of using this style: - Take more time to rewrite (which is not a good thing) - NOT HUNGARIAN NOTATION BUT YET THE MINIMAL VERSION - Your IDE is already good enough to deduce what type, if you hover the text (unless you use notepad or terminal console or Cult of Vi or Church of Emacs) - Explicit Heavy (break your fingers more)

Inspire by the following style guide: - Google C++ Style Guide

Inspire by the following programming language: - AngelScript - Rust - C++

It has been adopt in my own development, and in some public projects already. However this is still a stylize version.

If you use this and like it, please share to any fellow developers that may find good use of the knowledge. Please properly cite and credit to the original author.

Standard Rules

- How you say it naturally, should be named as that. DO NOT FORCE NAMING.
- Minimize standard as much as you can be.
- Please try to avoid naming things with s character, it really is an OCD issue many times
- Avoid many technical debt by being flexible.
- Delete your old and unused code, it's a debt. Archive it in txt file if needed.

Camelcase

- CamelCase
- SnakeCase

Header Comment

```
//
//(c) organization - name - year
//describe what your scripts do and other stuff
//
```

Comments

You should not spend much time in writing comments

Recommended in most use case :

```
// Comment goes here
```

```
/*
Paragraph comment goes here
*/
```

Header comment when need to seperate out of the above when use in conjunction:

```
//--[Variable]--
```

```
public class ExampleClassName {
  //--[Variable Header]--
  public int i32_something;
  //--[Function Header]--
  public Update(){
  }
  public int GetNumber() {
    return 0;
```

```
  }
}
```

For productivity boost, there is also intensity version if needed

```
// TODO: Task here

// PROBLEM Year.Month.Date : describe current problem, please add year month date in case yo

// PROBLEM 2023.10.11 : describe current problem, please add year month date in case you for

// BLAME: Insert someone name here on what basic, go to court regardless if they are right o
```

Productivity but intensity version:

```
// TODO TODO: Task here

// TODO TODO TODO: Task here

// TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TOD

// BLAME BLAME: Insert the nearest person name to you

// BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME BLAME
```

Constants

In a not unorganize way or you do not know how the structure is, it is highly recommended to stick with these 2 following:

ALL CAPS WITH UNDERSCORE

```
// default: 0
public static readonly int K_ENEMY_NUMBER = 0;
```

CamelCase Style

```
// default: 0
public static readonly int KEnemyNumber = 0;
```

In organize way, it can be written like this (although not recommended in most case)

```
// default: 0
public static readonly int K_EnemySpawner_EnemyNumber = 0;
```

Constant Class

Constant class should only ever be designed as database to get value in everywhere or only be in specific class.

```
public class KEnemyIndex {
  public static readonly int K_Grunt = 0;
  public static readonly int K_Boss_MachineGun = 1;
}

public class KEnemyUUID {
  public static readonly string K_Grunt = "1739198b-51bf-4e44-90b1-1772acfd49ef";
  public static readonly string K_Boss_MachineGun = "e56c9c80-3778-44f4-a6a0-41bdae4bb26b";
}
```

Enum

```
public enum KAIAnimation {
  K_Null = -1,
  K_None = 0,
  K_Idle,
  K_Run,
  K_AttackMelee,
  K_AttackRange,
}
```

Namespace

Namespace is highly recommended. It seperate your code from others.

```
namespace namespacegoeshere {

}
```

Variable Data Size

It is good to refresh your knowledge, there are limitation in computer hardware regarding calculation and runtime.

Variable Naming

Construction

For variable: - CollectionType_VariableType_VariableName - CollectionType_VariableName

```
int[] sz_i32_enemyNumber;
int[] sz_enemyNumber;
string str_characterName;
```

For variable (with pointer): - CollectionType_VariableType_pVariableName

EXAMPLE:

```
int* i32_pEnemyNumber;
CharacterSpawner* m_pSomething;
```

Public Variable

Public means public, by design it should mainly use in external class to reference variable & functions, rarely use in their own creation class

```
private int i32_number;
public int Number {
  get {
    return i32_number;
  }
  set {
    i32_number = value;
  }
}

private EnemySpawner[] sz_m_enemySpawner;
public EnemySpawner[] EnemySpawners {
  get {
    return sz_m_enemySpawner;
  }
  set {
    sz_m_enemySpawner = value;
  }
}
```

```
private int i32_number;
public void SetNumber(int arg_value) {
  i32_number = arg_value;
}
public int GetNumber {
  return i32_number;
}
```

Private Variable

This style highly recommended to just use VariableType_VariableName

```
private int i32_number;
private double[] sz_d_number;
```

However you can simply use C# Standard if needed, start with _

```
private int _i32_number;
private double[] _sz_d_number;
```

Array

Normal Array: - sz_somethingName - sz_type_somethingName

Array in Array: - szn_somethingName - szn_type_somethingName

List

Normal List: - list_something - list_type_something

```
List<float> list_f_allEntityHealth = new List<float>();
List<AudioClip> list_m_audioClip = new List<AudioClip>();
```

List in List in List: - listn_something - listn_type_something

```
List<GameAsset> listn_m_gameAssetDatabase = new List<GameAsset>();
List<AudioItem> listn_m_audioDatabase = new List<AudioItem>();
```

Boolean

- is (recommended)
- can
- has (use this only when need to check an item in array, or in certain suitable context)

When naming boolean, please keep the meaning positive to make it easier to manage this data variable

```
bool isTouchYet;
bool canJump;
bool IsWeaponInHand(){
  return result;
}
bool HasWeaponInArray(WeaponItem arg_weaponItem){
  return result;
}
```

Int

- i8
- i16 (above this is special for certain use case, custom hardware)
- i32 (highly recommended, common use)
- i64 (below this is special for certain use case, custom hardware)
- i128
- i256
- i512

```
int i32_variableName;
int i16_variableName;
```

Signed Int

- s8
- s16
- s32 (recommended, common use)
- s64
- s128
- s256
- s512

```
signed int s32_variableName;
```

Unsigned Int

- u8
- u16
- u32 (recommended, common use)
- u64
- u128
- u256
- u512

```
unsigned int u32_variableName;
```

Float

- f (recommended, common use)
- f32 (Rust style)
- f64 (Rust style)

```
float f_variableName;
```

Double

- d (recommended, common use)

```
double d_variableName;
```

String

- str

```
string str_variableName;
```

Char

- char

```
char char_variableName;
```

Char

- uchar

```
unsigned char uchar_variableName;
```

Vector

- vec2
- vec2Int
- vec3
- vec3Int

```
Vector2 vec2_enemyLocation;
Vector3 vec3_playerLocation;
```

Long

- l

```
long l_something;
```

Long Long

- ll

```
long long ll_something;
```

Long Long Double

- lld

```
long long double lld_something;
```

Custom Class

All custom class that was created by user or 3rdparty all start the following letter - m

```
EnemyManager m_enemyManager;
Enemy[] sz_m_allEnemy;
List<Enemy> list_m_allEnemy = new List<Enemy>();
```

Function Method Naming

```
public int SetNumber(int arg_index = 0, int arg_value) {
  //Content goes here
}
public int GetNumber() {
  //Content goes here
}
```

Function Parameter Naming

Each parameter require to be prefix with arg_

```
public void SetNumber(int arg_value) {
  value = arg_value;
}
```

Function Method Prefix

- Init
- Start
- Update
- Increase
- Decrease
- Add
- Remove
- Clear
- Sort
- Get
- New
- Create
- Destroy
- Set
- Reset
- On / Event
- Is
- IsEqual
- IsContain
- Has
- Routine / Schedule - for Async

For Loops

Alphabet, but not enforce - i > j > k > l > m > n > o > p

```
for(int i = 0; i < 0; i++) {
  for(int j = 0; j < 0; j++) {
    for(int k = 0; k < 0; k++) {

    }
  }
}
```

Interface Class Naming

Prefix start with letter I

```
public interface IClassNameHere {
  // Variable here
}
```

Abstract Class Naming

Prefix start with ABS

```
public abstract class ABSClassNameHere {
  // Variable here
}
```

ScriptableObject Class Naming

Prefix start with SO

```
public class SOClassNameHere : ScriptableObject {
  // Content goes here
}
```

Return

Return is a signal that escape out of the function

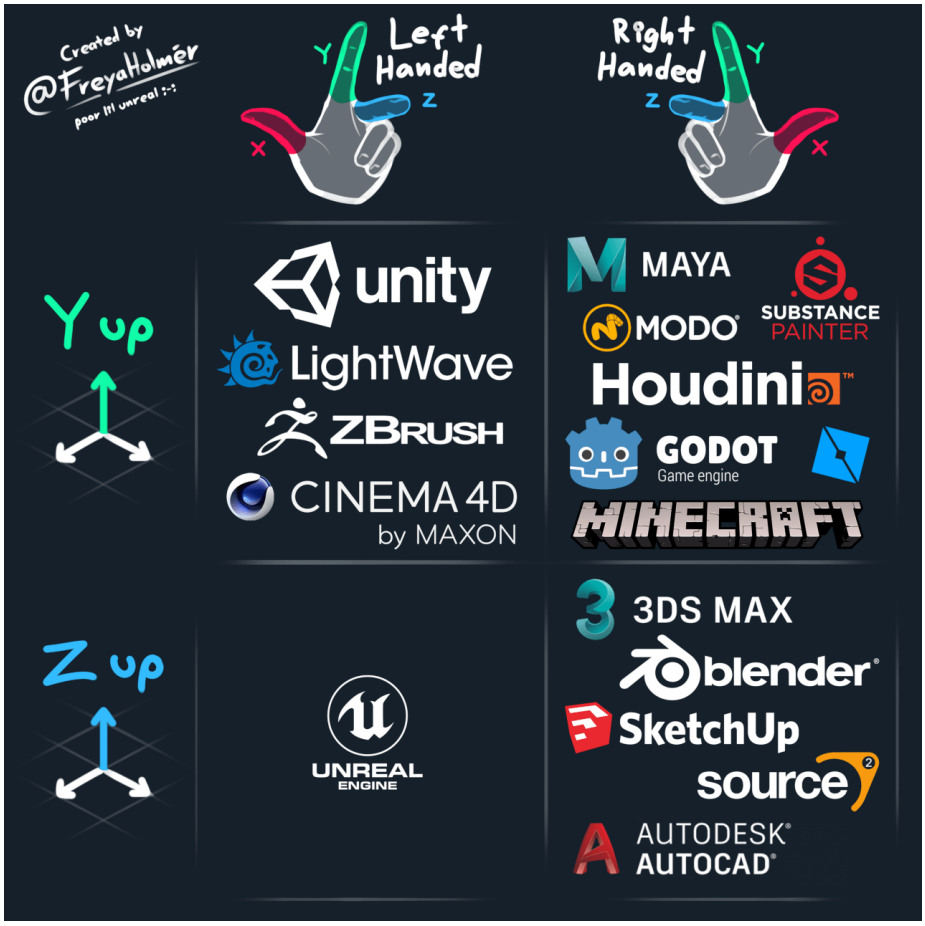
-> RECOMMENDED to write return in a seperate line, with brackets

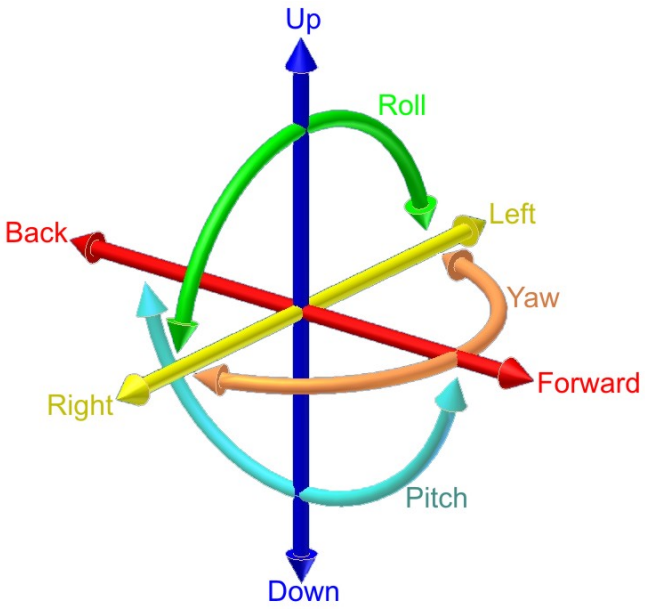
```
public int GetNumber() {
  return 0;
}
```

Considerations

It is a maddening process to say which is X Y Z in 3D Space

Please just use Roll Pitch Yaw as standard when talking to others regarding 3D Space





Bad & Removed Design

Please do not apply this, it will just make the rewriting more painful than it should be.

You should spent more times writing code than reading comments

```
// ISSUE: what is the issue here
// PROBLEM: what is the issue here

// DO NOT: will

// WILL NOT FIX: just use it regardless

// UNO: reverse what it is supposed to

//this style is too unnatural, uncanny, hard to deduce it is positive or not (bad design sy
bool bCanDone;
bool b_canBeDone;

int i_something; (confuse with interface class which has same prefix character i, hungarian

int int_something; (too redundant and why it is Hungarian Notation again)

float float_something (too redundant, and Hungarian Notation)

szxNumber and listxNumber (by context it means, sz * number, list * number)
- szx2
- szx3
- szx4
```

```
- listx2
- listx3
not useful to be this specific, it lead to massive rewrite (it already happen with my own pr

Add underscore to function name
int Get_FunctionName();

Please just follow the standards of C++ or C#
- C++: int getFunctionName();
- C#: int GetFunctionName();
```

Common Design Pattern

- Composition
- Builder
- Singleton

DO NOT IN ANY CIRCUMSTANCES

- Bad Naming Scheme

Feedback

- If there are something that doesn't make sense, reach out to me at contact information area.

Contact Information

- LinkedIn: www.linkedin.com/in/long-hoang-857578240
- Twitter: https://twitter.com/long_planner
- Email: hoanglongplanner@gmail.com

References

- https://docs.godotengine.org/en/stable/tutorials/3d/introduction_to_3d.html
- https://mastodon.social/@acegikmo/109429307211544506
- https://simple.wikipedia.org/wiki/Pitch,_yaw,_and_roll
- https://github.com/ziglang/zig/issues/6417
- https://softwareengineering.stackexchange.com/questions/102689/what-is-the-benefit-of-not-using-hungarian-notation
- https://www.w3schools.com/cs/cs_data_types.php