ZuluOneZero - Unity Style Guide

Take this example from the Unity API Manual. This is the best place to start. If it looks like code from the manual – then you got to be on the right track.

```
using UnityEngine;
using System.Collections;
// A grenade
// - instantiates an explosion Prefab when hitting a surface
// - then destroys itself

public class ExampleClass : MonoBehaviour
{
    public Transform explosionPrefab;

    void OnCollisionEnter(Collision collision)
    {
        ContactPoint contact = collision.contacts[0];
        Quaternion rotation = Quaternion.FromToRotation(Vector3.up, contact.normal);
        Vector3 position = contact.point;
        Instantiate(explosionPrefab, position, rotation);
        Destroy(gameObject);
    }
}
```

1. Comments make sense to humans and take the form of a). A statement of fact describing what this is (“a grenade”) then b). A brief description of how that action (behaviour) is implemented.
2. The Class is in PascalCase and is descriptive. Well “ExampleClass” is not a great example of this but “GrenadeExplosion” is.
3. Variables or fields are descriptive and in camelCase. Like “explosionPrefab” tells you it's a prefab and what it does.
4. Variables say what they are and common usage names are best. For example - contact, rotation, and position.
5. Curly braces are on the next line.
6. Line spaces used before methods or functions and most everything else in clear blocks.

Here is a few more guidelines along this path:

1. static variables or fields are written camelCase (don't capitalise there is no need to shout).
2. Use the default Code Editor (IDE) settings (four-character indents, tabs saved as spaces, etc).
3. Write only one statement per line.
4. Indent continuation lines (and make them regularly sized if possible).
5. Use parentheses to make clauses in an expression as obvious as possible.

Example: if ((var1 > var2) && (var1 > var3))

6. Don't use an underscore in-front of a variable unless you have passed it into a function where it remains the same.

```
float myFightingStrength;

TakeDamage(float _myFightingStregth, float _attackDamage)
{
        _myFightingStregth--;
        return _myFightingStregth;
}
```

File Naming Convention

1. No spaces on file or directory names.

2. Use PascalCase.

Folder Structure

These are the basic folders common to most projects:
Animation
Art (the 2D stuff)
Audio
Characters (I like to separate all my "player" stuff into one folder)
Objects (the raw 3D stuff – models/sculptures)
Prefabs
Scenes
Scripts

Assets

1. Use treeSmall (not smallTree). So you can group all tree objects together instead of all small objects.

2. Use descriptive suffixes instead of iterative: tank_burned not tank_02.

3. Use a leading underscore (_snake_case) to make important object instances stand out. Example: _bluePlayer

Models

Use Y up, -Z forward and uniform scale when exporting.