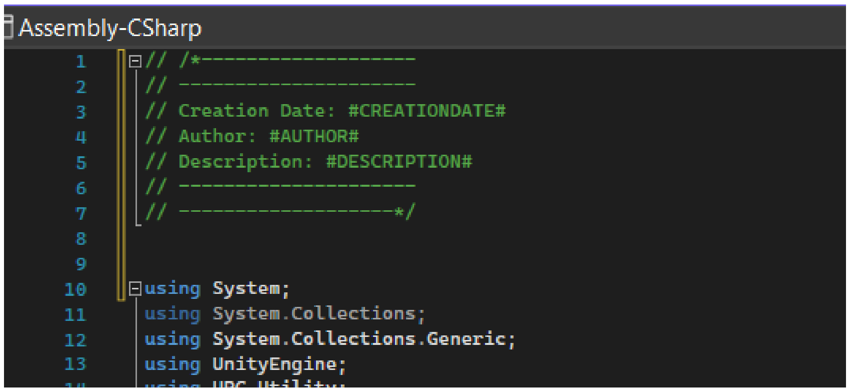
File structure

+ - - - Art
| + - - - Materials
| + - - - Models
| + - - - Textures
| + - - - Thirdparty
+ - - - Audio
| + - - - Music
| + - - - SFX
| + - - - Thirdparty
+ - - - Animation # Separate subfolders containing controller and clips
| + - - - Game
| + - - - UI
+ - - - Source
| + - - - Scripts
| + - - - Plugins
| + - - - Shaders
+ - - - Game
| + - - - Scenes
| + - - - Prefabs
| + - - - Resources
+ - - - Docs # Wiki, concept art, notes, this doc etc.

Coding style

Header comment



Assembly-CSharp

```
// /*---------------------
// ----------------------
// Creation Date: #CREATIONDATE#
// Author: #AUTHOR#
// Description: #DESCRIPTION#
// ----------------------
// ---------------------*/


using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

Purpose: Easily identify the use of the script and who made it

Naming conventions

```
// PascalCase for classes
public class GroundingReport
{
}

// PascalCase for structs
public struct PlayerData
{
}

// PascalCase for public fields
public bool IsReady;

// m_ + camelCase for private fields
private Rigidbody m_rigidbody;

// PascalCase for method signature, camelCase for method parameters
public float GetDistance(Vector3 fromPoint, Vector3 toPoint)
{
    // camelCase for local variables
    float calculatedDistance = Vector3.Distance(fromPoint, toPoint);

    return calculatedDistance;
}
```

Layout

- Only one statement per line
- Only one declaration per line
- Parenthesis to make clauses in expressions apparent, like this

```
// Correct
if ((val1 > val2) && (val1 < val3))
{
}

// Wrong
if (val1 > val2 && val1 < val3)
{
}
```

Commenting

- Public members should have XML comment
- Place comment at separate line, not end of line.
- Begin comment text with uppercase letter.
- End comment with a period.
- Add space after comment delimiter

```
// This is the beginning of the comment,
// and we end the comment here.

/// <summary>
/// This method initializes the necessary values for the script to work.
/// </summary>
public void Initialize()
{
}
```

Implicitly typed local variables

When the type of the **local** variable is obvious from the right side or when precise type is not important, use implicit typing

```
var message = "Clearly a string.";
var age = 42;
```

If the type is not clear from the right side, you need to write the type directly. Do not rely on variable name or method name to infer data type.

```
// Get children can mean different things, so this is not considered clear
// and therefore should be specified.
int count = SomeClass.GetChildren();
```

Delegates

If possible, use `Action<>` and `Func<>` instead of defining new delegates.

```
public Action OnDamageCallback;

public Func<Enemy> OnEnemyKill;
```

Version management (Github)

Only commit related changes

A commit should only be for one change. If you fixed two bugs, it should be two different commits.

Commit often

By committing the work often, it’s easier for other team members to work with the most recent version, and reduces chances of merge conflicts.

Test code before committing

Make sure that the code is tested and no side effects are produced from the commit. This can lead to problems down the road if not catched beforehand.

Write good commit messages

Generally, commit messages should be less than 50 characters. Description should be used to answer some general questions: Why change? How is it different from last change?