Kirklind Signature

KIRKLIND CODING STANDARDS

Change Log:

Date	Author	Changes
1/26/2025	Benton Wilson	Created the document from scratch. Added all ideas and work.

About this document:

This document holds the coding standards for the Kirklind Signature brand. In this you will find references to third party standards along with a variety of exceptions and additions highlighted for importance when it comes to the development of warehouse warriors.

Outside Coding Standards to Follow:

We are going to adhere to the Microsoft C# coding standards when in both realms of C# identifier names as well as C# coding conventions.

- Can be found here: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/identifier-names

We will also follow the standards used in the Unity documentation for naming and scripting.

- Those can be found here: https://unity.com/how-to/naming-and-code-style-tips-c-scripting-unity

Some Exceptions/ Additions:

Comments:

Every method that is not start or update should be preceded by a comment. Ex:

```
// Function that deals with character movement.
public void Movement() {
// Logic of the function
}
```

General Layout:

The order of the classes in the script should be based on visibility:

- **Public**
- **Protected**

- **Private**

Then methods should follow in this order:

- **Unity start method (if used)**
- **Unity update method (if used)**
- **Other Unity methods**
- **Custom Methods**

```
void Start()
{
/*Code to run on first frame*/
}
void Update()
{
/*Code to run every frame*/
}
void OnCollisionEnter2D(Collision2D collide)
{
/*Code to run on collision*/
}
public exampleClassMethod()
{
/*Example Class Method Code*/
}
```

Don’t Clog the Start/Update Methods:

Create outside functions that deal with some logic or action and then call them in the start/update. Avoid writing logic in the Unity provided methods.

```
void start()
{
    PlayerMovement();
}

void update()
{
    PlayerMovement();
}

void PlayerMovement()
```

```
{
    //Logic of player movement
}
```

Naming Conventions:

Use Pascal Casing convention when defining Classes or Methods. Ex:

```
    Public int PlayerHealthCalc() {
    // logic of the code
    }
```

If you are defining a constant variable use all upper cases. Ex:

```
Const int MAXVALUE = 100;
```

No Hungarian. Ex:

```
    private int someName; // Do
```

```
    private int m_someName, _someName; // Don't
    static int s_someName; // Don't
```

Functions:

Functions do things, repetition is not a good enough reason to extract code to a function. If code is repeated, extract it to a local function or lambda.

```
   var calculationHelper = (a, b, c) => a + b + c; // Do
```

```
    int CalculationHelper(int a, int b, int c) { // Don't
    return a + b + c;
    }
```

We also are going to follow the standard that all scripts should define one action/event only.

```
Example: A player movement and then click to shoot script should
both be separate and attached to the correct object individually
(High cohesion). Functions shall only be passed the specific
parameters that are needed to accomplish the task (low
coupling).
```

```
// A correct function setup

public bool IsPasswordValid(string password, int minLength)
{
    // Check if the password meets the minimum length
    if (password.Length < minLength)
    {
        return false;
    }

    // Check if the password contains at least one uppercase letter
    bool hasUppercase = password.Any(char.IsUpper);

    // Check if the password contains at least one lowercase letter
    bool hasLowercase = password.Any(char.IsLower);

    // Check if the password contains at least one digit
    bool hasDigit = password.Any(char.IsDigit);

    // The password is valid only if all conditions are met
    return hasUppercase && hasLowercase && hasDigit;
}
```

<u>File Heading:</u>

All files should have a description at the top. In this heading you should include Name, TL#, and what the file is doing. Also, if this class is an inheritance, you should outline this in the main file heading.

```
/* File Heading Sample

* Name: Benton Wilson
* Role: TL4
* Project: Warehouse Warriors

* This is a script that is going to deal with defining an example
class. It will showcase the *coding standards for our project.

*/
```