コーディング規約

内容

1.	ディレクトリ構成	2
	命名規則	
	コーディングスタイル	
4.	コメント	5

1. ディレクトリ構成

(青文字は確実に入れたいもの黄文字は例)

```
-Assets
-Scripts
                                   (スクリプト)
      ⊢Behav i our
                 (継承用スクリプト)
         ├<u>PlayerBehaviour</u> (プレイヤーの基本動作)
         └EnemyBehaviour (敵の基本動作)
                 (エフェクトプレハブへ適用するエフェクトスクリプト)
      Effect
        └PlayerJumpEffect (ジャンプエフェクト)
                 (管理スクリプト)
        LEffectManager (エフェクトマネージャー)
ene (各シーンスクリプト)
        ⊢TitleScene
                    (タイトルシーン)
         -GameScene1
                      (ゲーム中シーン)
         ∟Resu|tScene
                       (リザルトシーン)
                 (ユーザーインターフェイススクリプト)
      ⊢UI
         ├StageTimeLimitUI (タイムリミットUI)
        ∟HpUI
                       (残機 UI)
                 (汎用、有用スクリプト)
      Utility
         ∟GameTimer
                       (他スクリプト内で時間カウントする際に使う)
```

以上の構成で管理

- 新たなフォルダの追加や、各フォルダに新しいスクリプトファイルを追加していく
- Audio フォルダ、AudioManager などがあってもいいかもしれない

2. 命名規則

- 共通命名
 - ▶ クラス、メソッド、プロパティ、フィールド、変数などは英語を基本とする
 - ▶ 複数単語の場合、2つ目以降の単語の頭文字を大文字にする

private void MoveFunction ()

- 変数命名
 - フィールドは先頭に_(アンダーバー) つけ頭文字を大文字とするpublic int _ArmorPoint;//アーマーボイント
 - ▶ ローカル変数は頭文字を小文字とする

float maxAltitude = 100;//限界高度

- 定数命名
 - ▶ 定数(Const, Enum)はすべて大文字で単語間を_(アンダーバー)で繋ぐ
 - ▶ Enum のメンバも上記のルールに従う

```
private enum STATE_ENUM
{
    /// 〈summary〉何もない状態〈/summary〉
    NONE,
    /// 〈summary〉動作する状態〈/summary〉
    MOVE,
    /// 〈summary〉死亡状態〈/summary〉
DEAD,
```

- クラス、メソッド、プロパティ命名
 - ▶ クラス、メソッド、プロパティは頭文字を大文字とする

public bool IsDead

3. コーディングスタイル

- #Region を使う
 - ▶ 以下の順に分類し Visual Studio で折り畳み表示ができるようにする

```
□ public class PlayerBehaviour : MonoBehaviour

define

serialize field

public field

private field

Unity function

public function

private function
```

各 region の内容

Define

定数を入れる

Serialize field

[serialize field]を適用したフィールドを入れる

Public, Private field

それぞれのアクセス修飾子のフィールドを入れる

Property

プロパティ―を入れる

Unity function

Unity 独自に定義されているメソッドを入れる

- private void Start()
- private void Update()
- private void OnCollisionEnter2D(Collision2D collision)
- Public, Private function

それぞれのアクセス修飾子のメソッドを入れる

● できる限り private や public をつける

private void MoveFunction ()

● ブロックを作る中括弧と同じ行にプログラムを書かない

```
if (isBoost)
{
    acceleration = addBoostSpeed;
    maxSpeed *= boostMaxSpeed;
}
```

● マジックナンバーは原則使用せず定数を定義

4. コメント

- クラス、メソッド、プロパティ、フィールドにはコメントを入れる
- Summery コメントをクラス、メソッド、定数などで使う
 - ▶ (マウスで選択したときに表示されるコメント)

- paramコメントを引数で使う
 - ▶ (マウスで選択したときに表示されるコメント)

```
/// <summary〉
/// コリジョンが当たった瞬間毎に1回だけ呼ばれる処理
/// </summary〉
/// <param name="collision"></param>
⑤ Unity メッセージ | 0 個の参照
private void OnCollisionEnter2D(Collision2D collision)
```

● メソッドやプロパティ内部にも機能単位でコメントを入れる