脳筋プログラマーの自己紹介

名前：浜松廉斗

学校：福岡情報ITクリエイター専門学校

ポートフォリオ：https://x.gd/at8C6

言語経験年月

C++	：2年
GitHub	：2年
Unity/C#	：1年6ヶ月
HTML	：2ヶ月
PHP	：2ヶ月

GitHub

URL：Hmmtrnt (github.com)

QRコード：

チーム制作物の概要

※ドラゴンのスペルミスはタイトルの見栄えのため
わざとミスしてます。

作品名	DoragonHunter
ジャンル	アクションゲーム
製作期間	2023/10/09～2024/02/13
制作環境	Unity/C#/GitHub Desktop
制作人数	
プログラマー	1名
モーションデザイナー	1名
UIデザイナー	1名
担当箇所	プログラミング全部

制作作品　オブジェクトの状態制御(StateBase)

各状態クラスの遷移

挙動制御を行う抽象クラス、StateBaseを作成した。
StateBaseによって各状態を管理するクラス間の遷移をわかりやすくした。

StateBaseの引数

引数にひとつ前の状態、次に遷移する状態を
追加することで特定の状態遷移時にのみ行いたい
処理を追加できるようにした

次の状態に遷移する関数

```
/// <summary>
/// 状態遷移.
/// </summary>
/// <param name="nextState">次に遷移する状態</param>
88 個の参照
private void StateTransition(StateBase nextState)
{
    // 状態終了時.
    _currentState.OnExit(this, nextState);
    // 次の状態の呼び出し.
    nextState.OnEnter(this, _currentState);
    // 次に遷移する状態の代入.
    _currentState = nextState;
}
```

StateBase.cs

```
1   // 状態管理の抽象クラス
2
    99+ 個の参照
3   public abstract class StateBase
4   {
5       /*プレイヤー*/
6       /// <summary>
7       /// ステート開始時呼び出し
8       /// </summary>
9       /// <param name="owner">アクセスするための参照</param>
10      /// <param name="prevState">ひとつ前の状態</param>
        24 個の参照
11      public virtual void OnEnter(Player owner, StateBase prevState) { }
12      /// <summary>
13      /// Update
14      /// </summary>
15      /// <param name="owner">アクセスするための参照</param>
        23 個の参照
16      public virtual void OnUpdate(Player owner) { }
17      /// <summary>
18      /// FixedUpdate
19      /// </summary>
20      /// <param name="owner">アクセスするための参照</param>
        23 個の参照
21      public virtual void OnFixedUpdate(Player owner) { }
22      /// <summary>
23      /// ステート終了時呼び出し
24      /// </summary>
25      /// <param name="owner">アクセスするための参照</param>
26      /// <param name="nextState">次に遷移する状態</param>
        22 個の参照
27      public virtual void OnExit(Player owner, StateBase nextState) { }
28      /// <summary>
29      /// ステート遷移の呼び出し
30      /// </summary>
31      /// <param name="owner">アクセスするための参照</param>
        22 個の参照
32      public virtual void OnChangeState(Player owner) { }
33
```

制作作品　プレイヤー (内部クラス)

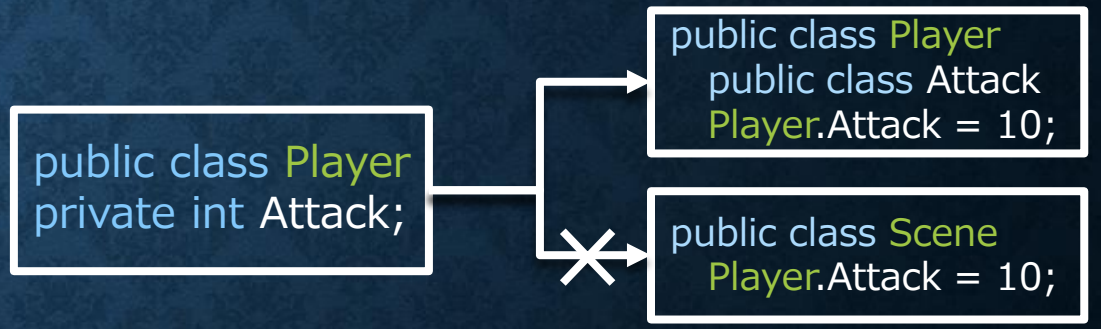
各状態クラスを内部クラスとして宣言

内部クラスにしたメリット

privateの変数や関数を各状態クラスのみに呼び出すことができ、
他のクラスに必要のない変数や関数を呼び出すことを防いだ。
そのため状態制御関係のバグの発生率が下がった。

内部クラスによるコードの肥大化

コードの肥大化がおこり、可読性の低下と
制作効率を損なうことになったのでpartial classを
使用することにした。




```
/*プレイヤー*/

using UnityEngine;

Unity スクリプト (3 件のアセット参照) 199+ 個の参照
public partial class Player : MonoBehaviour
{
    //--納刀状態--//
    private static readonly StateIdle               _idle = new();              // アイドル.
    private static readonly StateAvoid              _avoid = new();             // 回避.
    private static readonly StateRunning            _running = new();           // 走る.
    private static readonly StateDash               _dash = new();              // ダッシュ.
    private static readonly StateFatigueDash        _fatigueDash = new();       // 疲労時のダッシュ.
    private static readonly StateRecovery           _recovery = new();          // 回復.

    //--抜刀状態--//
    private static readonly StateDrawnSwordTransition _drawSwordTransition = new(); // 抜刀する.
    private static readonly StateIdleDrawnSword     _idleDrawnSword = new();    // アイドル.
    private static readonly StateRunDrawnSword      _runDrawnSword = new();     // 走る.
    private static readonly StateAvoidDrawSword     _avoidDrawnSword = new();   // 抜刀回避.
    private static readonly StateRightAvoidDrawSword _rightAvoid = new();       // 攻撃後の右回避.
    private static readonly StateLeftAvoidDrawSword _leftAvoid = new();         // 攻撃後の左回避.
    private static readonly StateSheathingSword     _sheathingSword = new();    // 納刀する.
    private static readonly StateSteppingSlash      _steppingSlash = new();     // 踏み込み斬り.
    private static readonly StatePiercing           _piercing = new();          // 突き.
    private static readonly StateSlashUp            _slashUp = new();           // 斬り上げ.
    private static readonly StateSpiritBlade1       _spiritBlade1 = new();      // 気刃斬り1.
    private static readonly StateSpiritBlade2       _spiritBlade2 = new();      // 気刃斬り2.
    private static readonly StateSpiritBlade3       _spiritBlade3 = new();      // 気刃斬り3.
    private static readonly StateRoundSlash         _roundSlash = new();        // 気刃大回転斬り.
```

```
/*待機状態*/

using UnityEngine;

Unity スクリプト 199+ 個の参照
public partial class Player
{
    2 個の参照
    public class StateIdle : StateBase
    {
        3 個の参照
        public override void OnEnter(Player owner, StateBase prevState)
        {
            // アニメーション開始.
            owner._idleMotion = true;

            if(prevState == _sheathingSword)
            {
                owner._motionFrame = 0;
            }
        }
```

制作作品　簡易クラス図(プレイヤー状態制御)