作品名：DoragonHunter

敵の様々な攻撃を回避しつつ
コンボ攻撃を叩き込め！

ジャンル:モンハン風**狩猟**アクション！

制作者：浜松 廉斗

大食いプログラマー浜松廉斗

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

GitHubURL：https://github.com/Hmmtrnt

言語経験年月
Unity/C#：2年、C++：2年等

ゲームループ

より高い難易度に挑戦！タイムを縮めよ！

クエスト受注

いざ出発！

モンスター討伐！

討伐成功！

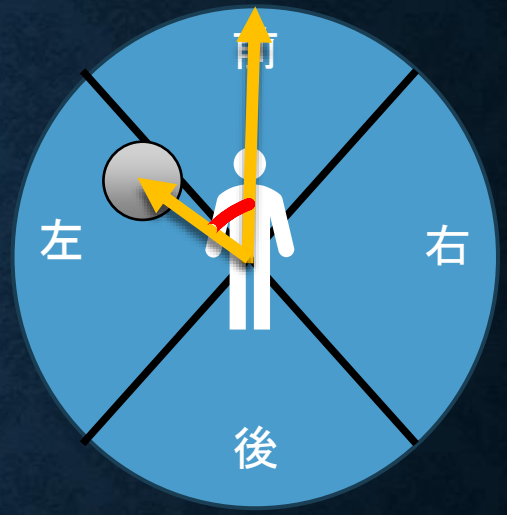
ランク更新！

未知の難易度に挑戦！

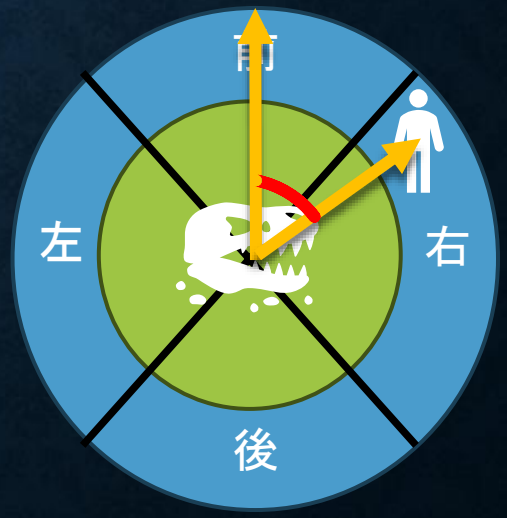
制作作品　ベクトルの活用

なす角で方向を求める

キャラクターの正面ベクトルとキャラクターから見て
スティックを傾けたベクトルの**なす角**を求めた。
更に正面ベクトルから見て右に傾いていたらなら正、
左に傾いていたら負のなす角を求めるようにしています。
そのなす角を求めることによってキャラクターから見て
前後左右どちらに回避するかを判断できるようにしている



モンスター側でも実装を行っている。
角度によって次に行う**攻撃パターンを絞り込む**ことが出来た。



更にプレイヤーの位置とモンスターとの位置の2点間の距離を
求めることで**遠距離攻撃かどうかを絞り込む**ことが出来た。


…プレイヤーの位置


…モンスターの位置


…スティックの位置

制作作品　各状態の経過時間を管理

経過時間によって**モーションキャンセル**を行う

このゲームはプレイヤーの攻撃の後隙は全体的に長くなっている。
しかし、攻撃の後隙に一部の入力を行うことで (※1)**モーションキャンセル**を行う。
モーションキャンセルが出来るのは攻撃コンボをつなげる時と回避を行う際に実行される。
このモーションキャンセルは現在の状態の経過時間を取得し、ある程度時間がたつと
次の行動の入力を行えるようにしている。
経過時間の計測は常に経過時間を計り続け、状態遷移を行った際に経過時間の値を
リセットしている。
さらにこの経過時間の値は効果音や当たり判定の発生にも活用している。

攻撃開始　　後隙　　モーションキャンセル

※1**モーションキャンセル**：現在のモーションを中断し次の行動に移ること

制作作品　状態制御(ステートマシン)

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。

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フィールドの変数や関数を呼び出すことが出来る。
そのため、他のクラスに必要のない変数や関数を呼び出しを
防ぎ、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。

StateBase.cs

```
// 状態管理の抽象クラス

99+ 個の参照
public abstract class StateBase
{
    /*プレイヤー*/
    /// <summary>
    /// ステート開始時呼び出し
    /// </summary>
    /// <param name="owner">アクセスするための参照</param>
    /// <param name="prevState">ひとつ前の状態</param>
    24 個の参照
    public virtual void OnEnter(Player owner, StateBase prevState) { }
    /// <summary>
    /// Update
    /// </summary>
    /// <param name="owner">アクセスするための参照</param>
    23 個の参照
    public virtual void OnUpdate(Player owner) { }
    /// <summary>
    /// FixedUpdate
    /// </summary>
    /// <param name="owner">アクセスするための参照</param>
    23 個の参照
    public virtual void OnFixedUpdate(Player owner) { }
    /// <summary>
    /// ステート終了時呼び出し
    /// </summary>
    /// <param name="owner">アクセスするための参照</param>
    /// <param name="nextState">次に遷移する状態</param>
    22 個の参照
    public virtual void OnExit(Player owner, StateBase nextState) { }
    /// <summary>
    /// ステート遷移の呼び出し
    /// </summary>
    /// <param name="owner">アクセスするための参照</param>
    22 個の参照
    public virtual void OnChangeState(Player owner) { }
```

```
/*プレイヤー行動全体の管理*/

using UnityEngine;

Unity スクリプト (3 件のアセット参照)|99+ 個の参照
public partial class PlayerState : 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.
```

```
/*待機状態*/

using UnityEngine;

Unity スクリプト|99+ 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;
            }
        }
```