

# ポートフォリオ

HAL名古屋  
ゲーム4年制学科  
澤村瑠人

# 目次

- 2p : 目次
- 3p : 自己紹介
- 4p ~ : 作品紹介
  - Knight Wanderer
  - キリトリセン
  - スタンピットヒーロー

# 自己紹介



名前 : 澤村瑠人  
生年月日 : 2001/06/03  
学校名 : HAL名古屋  
出身地 : 愛知県名古屋市

## 特技・趣味

読書が趣味で月に3～5冊の本を読んでいます。  
昔から虫や恐竜などの生物が好きで図鑑を読み始めたことをきっかけになりました。

## 資格・受賞歴

2022/12 CGエンジニア検定エキスパート  
2022/12 ゲームクリエイター甲子園 ユーザー大賞第3位  
2022/11 ゲームクリエイター甲子園 11月月間ツイート賞  
2022/10 J検 情報システム試験 基本スキル  
2021/12 画像処理エンジニア検定ベーシック  
2021/07 J検 情報活用試験 1級

## 使用している言語・ソフト

- ・ Visualstudio2017/DirectX11/C/C++
- ・ Unity/C#
- ・ GitHub (GitHub Desktop)
- ・ 1BITDRAGON
- ・ Blender

## 自己PR

私の強みは責任感があるところです。  
高校では空手となぎなたの修行をしながら、体育祭、文化祭の運営など学校行事にも力を入れていました。  
常に複数のタスクを抱えた生活は大変でしたが、その中で物事を投げ出さない責任感を身に着けました。  
現在も学校内組織に所属し、体験入学などの手伝いを行っています。

## 作品のURL

[https://drive.google.com/drive/folders/1QA7WctAYbvYAg-VBQjltVtDutiNAC9p?usp=share\\_link](https://drive.google.com/drive/folders/1QA7WctAYbvYAg-VBQjltVtDutiNAC9p?usp=share_link)

# Knight Wanderer



## 作品概要

キャラクターの行動タスクの処理を重視して制作を行っているアクションゲームです。

プレイヤーの行動に対してのリアクションにもこだわり、一本のバトルアクションゲームとしての表現をクオリティ高く制作しています。

タイトル : Knight Wanderer

ジャンル : バトルアクション

制作期間 : 1ヶ月(現在)

制作者 : 自分

使用した言語・ツール :

- C++
- DirectX11
- VisualStudio2017
- 1BITDRAGON

# Knight Wanderer

## 4月までに実装予定のもの

- ・ 群体制御
- ・ エフェクトや演出面の完全実装
- ・ UIの完全実装
- ・ リアクションオブジェクト実装

## 完成までに実装予定のもの

- ・ 高さによる行動判断
  - ・ 敵キャラクター複数時の特殊処理
  - ・ ゲーム本編外のリアクションの作成
- ※岩を切るとはじかれるなど



## 現在実装できているもの

- ・ 一通りのキャラクターの基本行動の実装
- ・ 条件達成時のシーン遷移
- ・ ライフの処理
- ・ 当たり判定
- ・ カメラワーク
- ・ 仮のBGM作成・実装

# キリトリセン



## 作品概要

この作品はハサミの”チョキチョキ”と”スーツ”という物を切る感触に注目して制作したゲームです。また、小学校を舞台にしており、ハサミをよく使ったあの頃のワクワクを思い出させるような雰囲気を作り出しています。

タイトル：キリトリセン  
ジャンル：パズルアクション

制作期間：4ヶ月

制作人数：14人

担当箇所：

- ・ BehaviorTreeを用いたAIによるタスク処理

使用した言語・ツール：

- ・ Unity/C#
- ・ GitHub Desktop
- ・ Visualstudio2017

## キリトリセン動画のURL

[https://www.youtube.com/watch?v=TyeUFf1\\_BXl&t=1s](https://www.youtube.com/watch?v=TyeUFf1_BXl&t=1s)

# キリトリセン

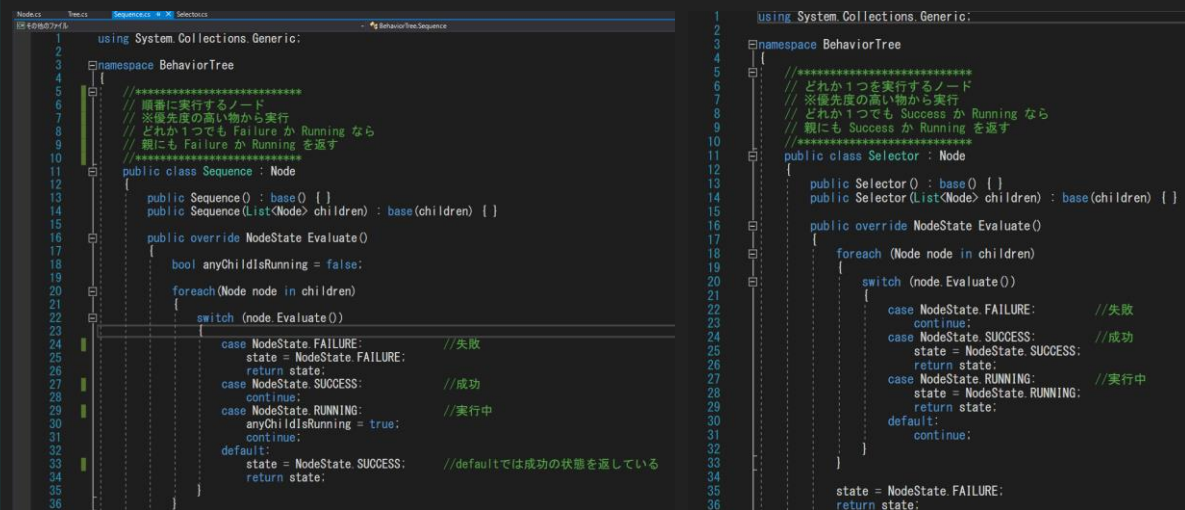
## アピールポイント

この作品における私のアピールポイントは BehaviorTree をスクリプトで一から構築し、動作までさせたことです。キャラクターの挙動管理に興味があり、今回の作品で AI のタスクの判断までの流れを学びながら制作しました。

## 苦労した点

AI のタスクを判断するプログラムの流れを理解することに苦労しました。  
内容理解のために海外の説明動画などを翻訳しながら制作を行っていたのでかなり時間もかかりました。

## 担当した箇所



```
using System.Collections.Generic;

namespace BehaviorTree
{
    // *****実行するノード*****
    // ※優先度の高い物から実行
    // どれか1つでも Failure か Running なら
    // 親にも Failure か Running を返す
    // *****
    public class Sequence : Node
    {
        public Sequence() : base() {}
        public Sequence(List<Node> children) : base(children) {}

        public override NodeState Evaluate()
        {
            bool anyChildIsRunning = false;
            foreach(Node node in children)
            {
                switch (node.Evaluate())
                {
                    case NodeState.FAILURE: //失敗
                        state = NodeState.FAILURE;
                        return state;
                    case NodeState.SUCCESS: //成功
                        continue;
                    case NodeState.RUNNING: //実行中
                        anyChildIsRunning = true;
                        continue;
                    default:
                        state = NodeState.SUCCESS; //defaultでは成功の状態を返している
                        return state;
                }
            }
        }
    }
}

using System.Collections.Generic;

namespace BehaviorTree
{
    // *****実行するノード*****
    // ※優先度の高い物から実行
    // どれか1つでも Success か Running なら
    // 親にも Success か Running を返す
    // *****
    public class Selector : Node
    {
        public Selector() : base() {}
        public Selector(List<Node> children) : base(children) {}

        public override NodeState Evaluate()
        {
            foreach (Node node in children)
            {
                switch (node.Evaluate())
                {
                    case NodeState.FAILURE: //失敗
                        continue;
                    case NodeState.SUCCESS: //成功
                        state = NodeState.SUCCESS;
                        return state;
                    case NodeState.RUNNING: //実行中
                        state = NodeState.RUNNING;
                        return state;
                    default:
                        continue;
                }
            }
        }
    }
}
```



# スタンプットヒーロー



## 作品概要

このゲームはステージ内を縦横無尽に飛び回りながら、敵を倒すという爽快感を追求したゲームです。敵や障害物を破壊しながらゲームを進めるため、足場も破壊してしまうので爽快感だけでなく思考要素も必要となります。

タイトル：スタンプットヒーロー

ジャンル：パズルアクション

制作期間：4ヶ月

制作人数：9人

担当箇所：

- ・フィールド作成処理
- ・タイマー処理
- ・クリア時のポーズ機能

使用した言語・ツール：

- ・C++
- ・DirectX11
- ・GitHub
- ・Effekseer
- ・Visualstudio2017



# スタンプットヒーロー

## アピールポイント

この作品ではステージごとのフィールド作成処理を力を入れて作成しました。初めてのチーム制作の作品でほかのチームメンバーが処理を追加しやすいようにコメントなどを見やすくすることを意識して制作しました。

## 苦労した点

- 1.企画からプログラムの設計までほとんどすべての工程にかかわり、処理するタスクそのものが多かったこと。
- 2.自身が作成した敵キャラクターが作成終盤でなくなったこと。



```
471 // 配列の中身の処理
472 // 二次元配列Map内で、"OO"の場所に描画する
473 // 括弧内の数字を呼び出す
474 for (int Height = 0; Height < MAP_HEIGHT; Height++)
475 {
476     for (int Width = 0; Width < MAP_WIDTH; Width++)
477     {
478         // #pragma region 呼び出されるオブジェクト情報
479         switch (g_Map[eArea][Height][Width])
480         {
481             // *****
482             // case文の中に描画したいオブジェクト情報を書いてください
483             // *****
484             // 何もしない
485             case 0: {
486                 // 何もしない
487                 break;
488             }
489             // 通常Y_2
490             case 1: {
491                 // マップチップ"1"の場所に描画するもの
492                 // 通常ブロック
493                 SetBlock(XMFLOAT3(g_MapPosOrizin.x + (Width * BlockSize.x),
494                     g_MapPosOrizin.y - (Height * BlockSize.y) * 2,
495                     g_MapPosOrizin.z), false, XMFLOAT2(0.5f, 2.0f), XMFLOAT2(1.0f, 11.0f));
496                 break;
497             }
498             // 通常Y_3
499             case 2: {
500                 // マップチップ"2"の場所に描画するもの
501                 // 無敵ブロック
502                 SetBlock(XMFLOAT3(g_MapPosOrizin.x + (Width * BlockSize.x),
503                     g_MapPosOrizin.y - (Height * BlockSize.y) * 2,
504                     g_MapPosOrizin.z), false, XMFLOAT2(0.5f, 3.0f), XMFLOAT2(1.0f, 1.0f));
505                 break;
506             }
507         }
508     }
509 }
```