

# 作品集

---

日本電子専門学校 ゲーム制作研究 平島 迅

# 自己紹介

氏名 平島 迅

年齢 20歳 (2004年 11月 3日)

所属 日本電子専門学校 ゲーム制作研究科

出身 長野県 茅野市

趣味 音楽鑑賞・作成 読書



# 自己紹介：趣味について

## 音楽鑑賞・作成

よく聴いているジャンルはボーカロイドで、最近ではクラシック音楽を聴く事にハマっています。お気に入りにはロウワーとカノンです。

作曲はDTMでBGMなどを作っています。最近ではチャーチモードを用いて作っています。

## 読書

主に読んでいるのは音楽理論や物理学関連の専門書などです。

作曲したもの

github



<https://github.com/Hirasima/Music.git>

GoogleDrive



[https://drive.google.com/drive/folders/1DqGPEdWuivygZFhF1nTUpjRLawIL\\_OgUg?usp=drive\\_link](https://drive.google.com/drive/folders/1DqGPEdWuivygZFhF1nTUpjRLawIL_OgUg?usp=drive_link)



Stage1

Reset LB Menu RB

TIME 165

# 使用経験のある言語・ソフトウェア

## ・プログラミング言語

C	2年5か月
C++	2年5か月
C#	1年5か月

## ・ソフトウェア

Unity	1年5か月
UnrealEngine	1年5か月
VisualStudio	2年5か月
Maya	1年
Cubase	5年1か月



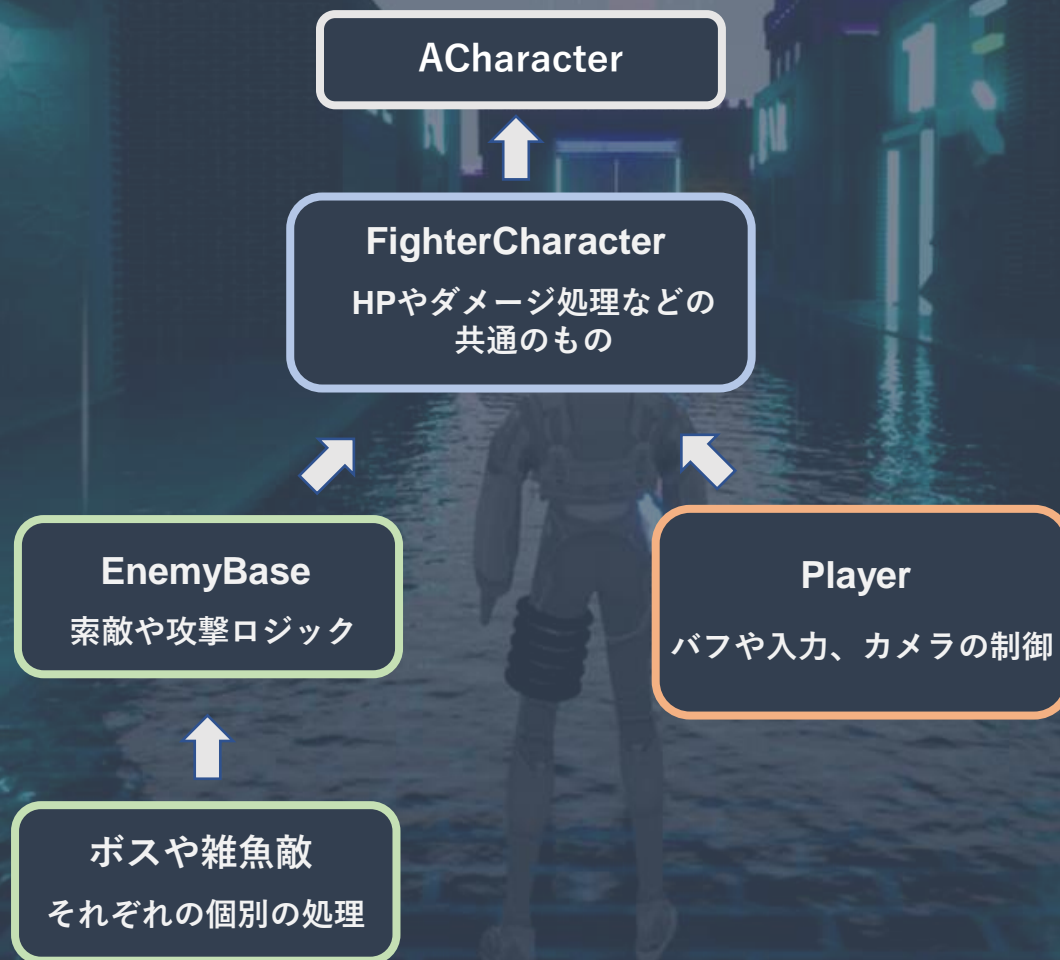


3年次 前期 制作  
TGS出展作品



# プレイヤーと敵の親クラスの実装について

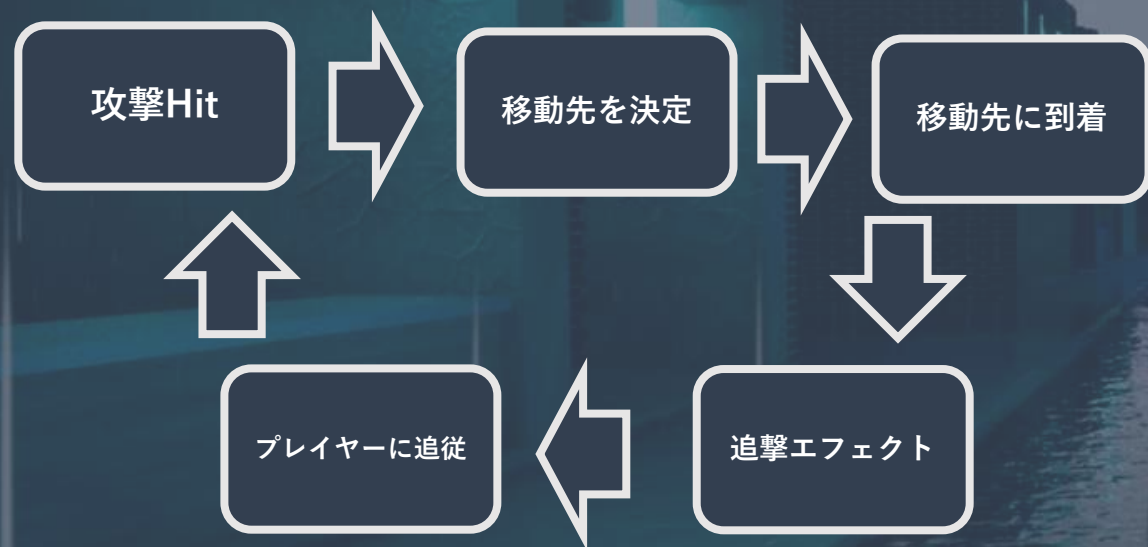
プレイヤーと敵の親クラスを実装することで、  
攻撃、被攻撃などの処理やHPなどの変数をまとめて実装することができ、  
修正などもし易いように制作しました。





# 追加ダメージバフ時の追撃ビットについて

プレイヤーの攻撃時にランダムに決めた  
Pitch(仰角)とYaw(縦軸回転)と敵座標から攻撃位置を決めて  
そこに向けて線形補間する事で、ビットの動きを作りました



//回転とターゲット在庫座標から攻撃位置を決める

```
attackPos.X = AttackRadius * FMath::Cos(FMath::DegreesToRadians(attackRot.Pitch)) * FMath::Cos(FMath::DegreesToRadians(attackRot.Yaw));  
attackPos.Y = AttackRadius * FMath::Cos(FMath::DegreesToRadians(attackRot.Pitch)) * FMath::Sin(FMath::DegreesToRadians(attackRot.Yaw));  
attackPos.Z = AttackRadius * FMath::Sin(FMath::DegreesToRadians(attackRot.Pitch));
```

三角関数を用いてターゲットを中心とした半球状の位置から攻撃する場所を計算



# アニメーション制御について

歩きや待機など**継続**して流すものと攻撃や回避などの**単発**で流すもので分けて制御しました



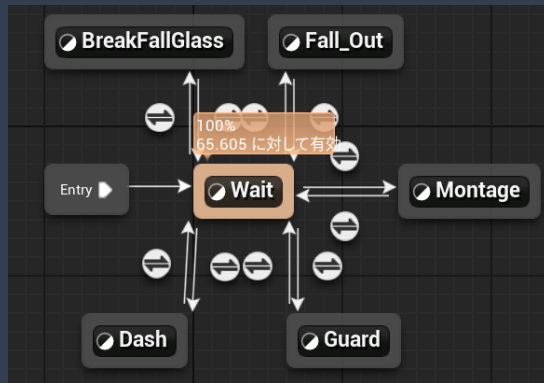
プレイヤーの  
ステータス



アニメーションBP



アニメーション再生



プレイヤーの状態をアニメーションBPに渡し  
アニメーションシーケンスを流す

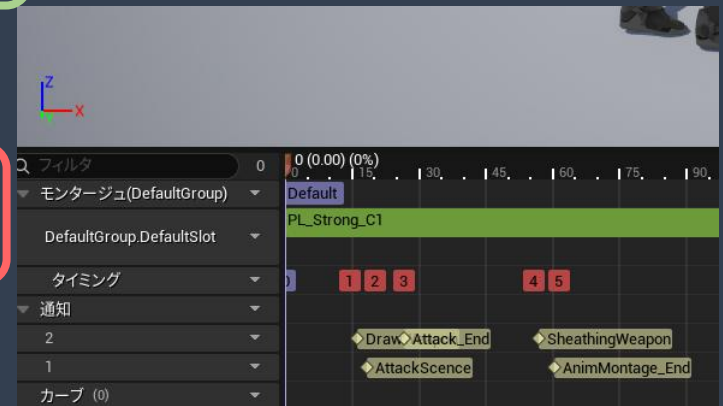
アニメーション再生



AnimNotify



アニメーションに合わせた  
処理






プレイヤーのPlayAnimMontage関数で再生し  
AnimNotifyで当たり判定などを制御

# プロジェクトを通してこだわった点

プランナーの方がスクリプトを触らずに攻撃力やアニメーションを変更しやすくなる事を  
目指して制作しました

また攻撃を統一規格で管理する為に  
アニメーションや攻撃力、  
よろけの強さなどをまとめた構造体を作り  
管理し易くなるよう工夫しました。

Attack Ups	5 配列エレメント	+	🗑	↔
▼ インデックス [ 0 ]	8 メンバー	▼		↔
Anim Mantage	 PL_Strong_C1_Montage	🔄	🔍	↔
Attack Point	30			↔
Knock Back Power	0.0			↔
Guard Break Power	0			↔
Stag Type	None	▼		↔
Can Knock Up	<input type="checkbox"/>			↔
▶ Area Collision Relative Location	100.0	0.0	0.0	↔
▶ Area Collision Size	100.0	100.0	100.0	↔
▼ インデックス [ 1 ]	8 メンバー	▼		↔
Anim Mantage	 PL_Strong_C2_Montage	🔄	🔍	↔
Attack Point	25			↔
Knock Back Power	0.0			↔
Guard Break Power	0			↔
Stag Type	None	▼		↔
Can Knock Up	<input type="checkbox"/>			↔
▶ Area Collision Relative Location	200.0	0.0	0.0	↔
▶ Area Collision Size	100.0	400.0	100.0	↔
▼ インデックス [ 2 ]	8 メンバー	▼		↔
Anim Mantage	 PL_Strong_C3_Montage	🔄	🔍	↔
Attack Point	40			↔
Knock Back Power	0.0			↔
Guard Break Power	0			↔
Stag Type	None	▼		↔
Can Knock Up	<input type="checkbox"/>			↔

Play Pawn				↔
Default Camera FOV	90.0			↔
▶ Spring Arm Offset	0.0	0.0	93.0	↔
Default Spring Arm Length	450.0			↔
Camera Pitch Max	80.0			↔
Camera Pitch Min	-30.0			↔
Lock on Pitch	-20.0			↔
Lock on Lerp Alpha	1.0			↔
Camera Reset Speed	150.0			↔
Lock on Distance	2000.0			↔
Lock on Move Speed Correction Value	0.6			↔
PCTurning Speed	5.0			↔
Camera Rotation Acceleration	10.0			↔
Camera Rotation Deceleration	10.0			↔
Camera Rotation Speed Max	150.0			↔
Avoidance Move Speed	2000.0			↔
Buff Rimit Num	5			↔
Buff Duration by Under Attack	2.0			↔
Buff Duration Rate by Up Attack	2.5			↔
Buff Duration Stop Time	3.0			↔
Combo Buff Rimit Num	4			↔
Combo Buff Rate	1.5			↔
Speed Up Rate	1.12			↔
Guard Power Max	100			↔
Guard Power Increase Per Second	25.0			↔
Is Undead Mode	<input type="checkbox"/>			↔
Time Taken to Dead	5.0			↔
▶ Respawn Point	-10233.093824	58.327463	91.583903	↔
Attack Collision Hidden in Game	<input checked="" type="checkbox"/>			↔
Attack Actor Class	MyAttackActor	🔄	🔍	↔

ディメンショナル エニグマ

# DIMENSIONAL ENIGMA

2年次 後期 制作作品

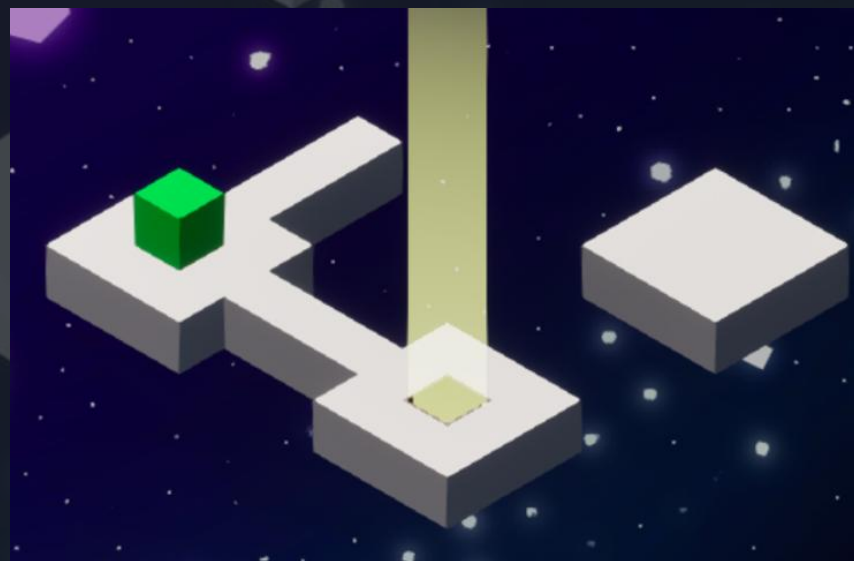
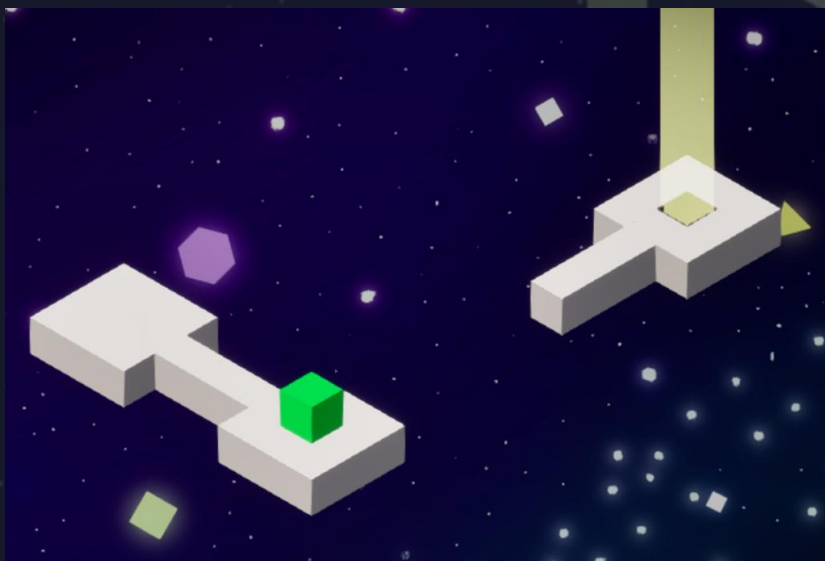


# ゲーム概要

作品名	Dimensional Enigma (ディメンショナル エニグマ)
製作期間	3ヶ月間 2024年 11月 ~ 2025年 2月
製作人数	6人 (プランナー：1名 デザイナー：1名 プログラマー：4名)
開発環境	UnrealEngine5.3
使用言語	C++
主な担当箇所	カメラ リザルト ステージ管理クラス

# ゲーム概要

錯視を利用してゴールを目指すパズルゲーム

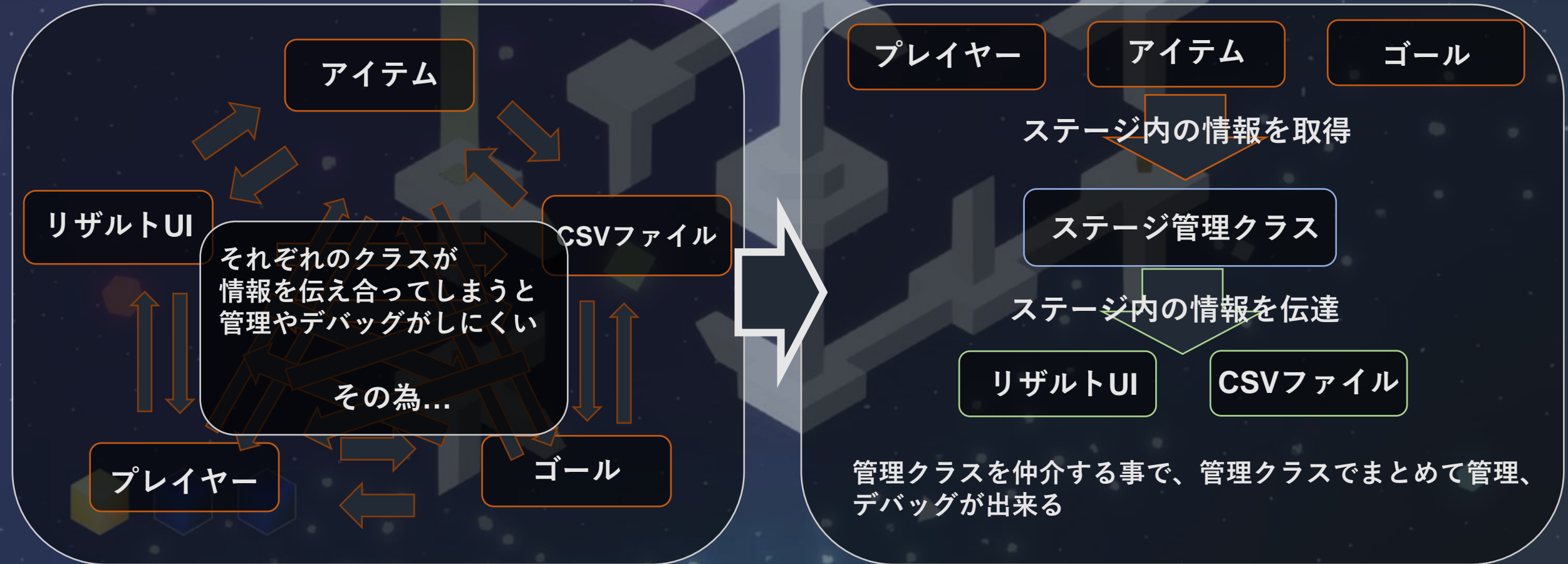


カメラを90度回転する事で、

ゴールと繋がる

# ステージ管理クラスについて

UEのをSubsystemを派生したステージ(Level)ごとの管理するクラスを作成し、ステージ内のオブジェクト同士の直接の参照を避けることで、セキュリティ向上を目指しました





# ランキング・ランク機能の実装について

ゲームを一度閉じたとしてもデータを保持したかった為  
また、実行ファイルにした後に変更が容易になるように  
CSVファイルを用いて製作



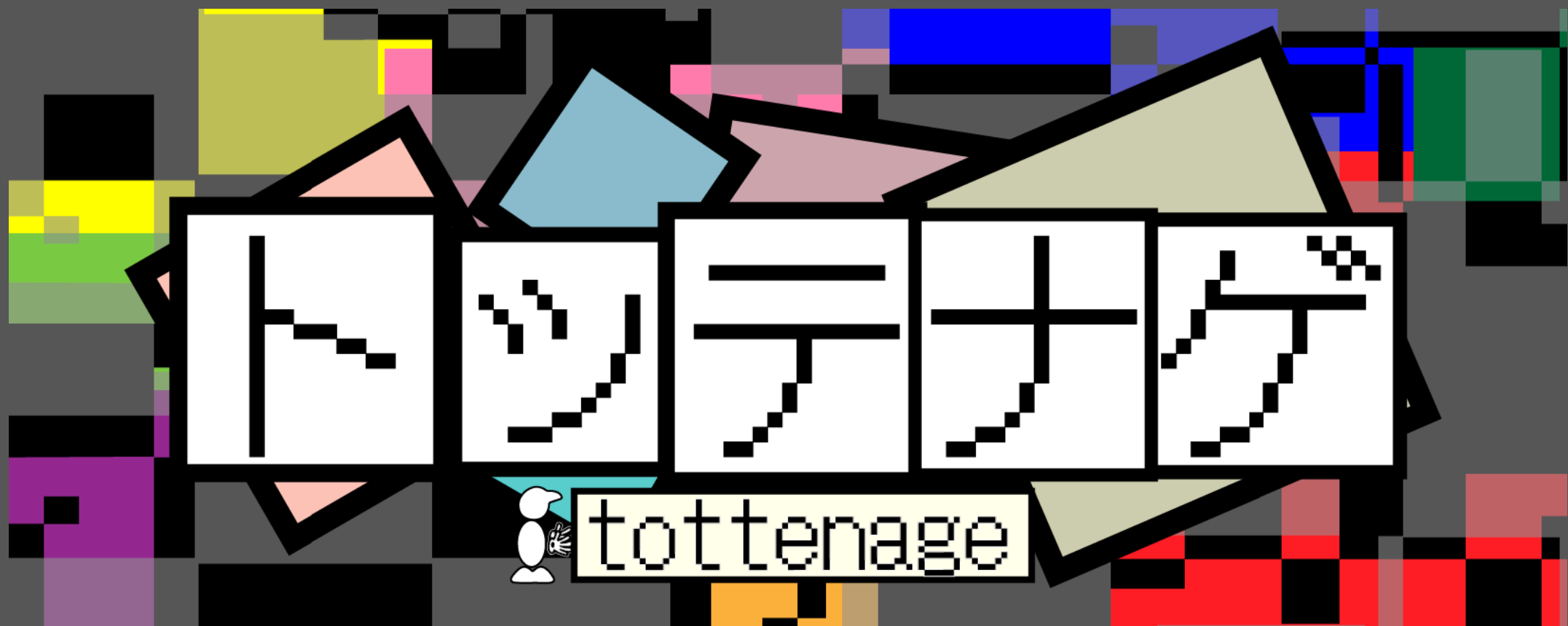
RankingData

- ・リザルト時にクリアタイムや歩数などを書き込む
- ・ランキングデータを読み込み表示
- ・書き込み時ファイルが存在しなかった場合、新たにファイルを作成する



QuotaData

- ・秒数、歩数、カメラ回転回数のノルマを書き込んでおくことで、  
リザルト時の結果と比べてランク付けをする

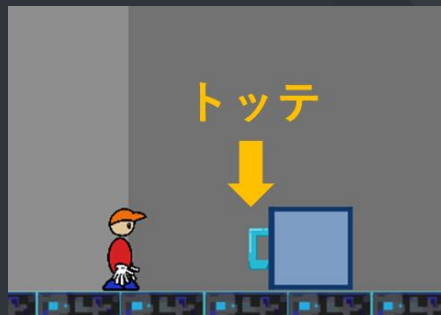


2年次 前期 制作作品

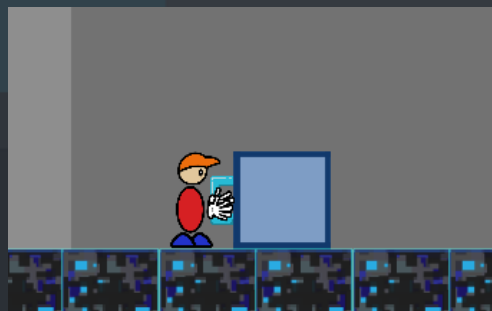




トッテを利用して鍵を取り時間内にゴールするゲーム



トッテの付いたものを



掴んで移動したり



投げる事が出来る



鍵を拾って



ゴールを目指す

Stage 1

Reset LB Menu RB



165

## ブロックの実装について

ブロックの状態を 置かれている・掴まれている・投げられている  
の3種類に分類して管理しました。

置かれている時は重力により落ちるが、  
横には動かない

掴まれている時はプレイヤーの動きに合わせて動く

投げられている時は加速度を持ち、  
動かなくなったら置かれている状態になる

# プレイヤーとブロックについて

プレイヤーの操作の受付からブロックの挙動までを  
3つのスクリプトにそれぞれオブジェクトの機能に分け実装しました



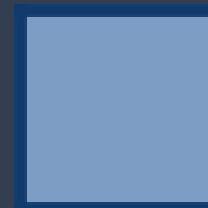
## PlayerBlockInteraction.cs

- ・プレイヤーからブロック・トッテへの干渉について
- ・プレイヤーにアタッチ
- ・プレイヤーの操作をトッテに伝える役目



## Handle.cs

- ・トッテの挙動について
- ・トッテにアタッチ
- ・プレイヤーとブロックとの仲介役
- ・掴める時にUIを表示



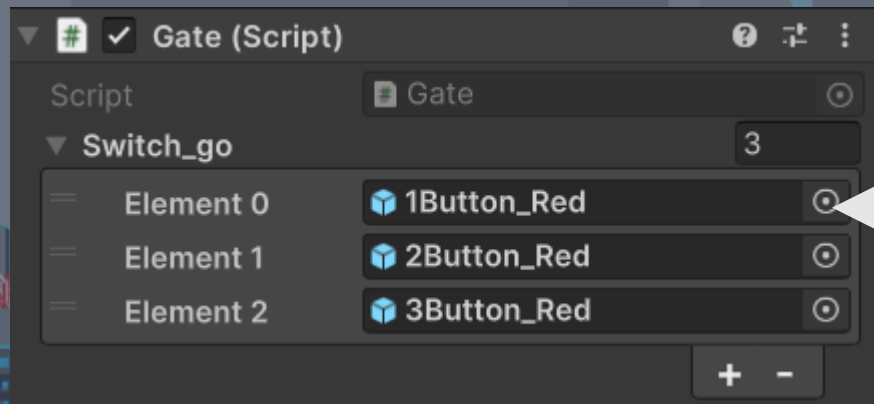
## Block.cs

- ・ブロックの挙動について
- ・ブロックにアタッチ
- ・ブロックの挙動を制御
- ・トッテからの操作により状態を変更

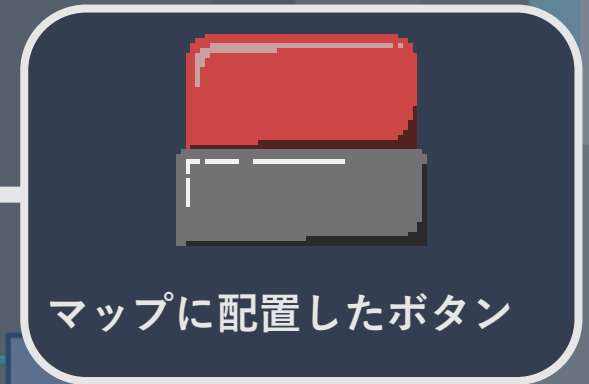


## ボタンとゲートの実装について

ゲートに対して複数のボタンを押すことで開くような仕様になっており  
エディタでステージに楽に実装できるように工夫しました



リンクさせたいゲートの  
スクリプトに登録



配列の数をマップに配置する際に自由に決められるよう制作しました