Monster Coliseum
モンスターコロシアム

《モンスターコロシアム》作品紹介

制作者：

何　兆祺（カ　チョウキ、Ho Siu Ki）

日本工学院八王子専門学校
ゲームクリエイター科四年制　ゲームプログラマーコース
2020 年 3 月　卒業見込み

目次

作品概要

ジャンル：	3D アクション
使用プログラミング言語：	C++
開発環境：	DX ライブラリ (https://dxlib.xsrv.jp/index.html)
制作期間：	約 5 か月（2019 年 2 月～6 月）

制作動機

この作品を制作した理由は、ゲームエンジンに頼らずに C++で 3D アクションゲームの制作に挑戦したかったからです。

私は 3D アクションゲームが好きで、そのプレイヤーキャラクターの挙動制御に非常に興味を持っています。専門学校に在籍する 4 年間、3D ゲーム制作に必要なスキルを身につける目標を立てましたが、最初の 2 年は C++と数学の知識が足りなかったため、Unity の機能を利用して仕様を実装することに止まりました。

3 年次の後期になってから、本格的に C++を扱い始め、かつ Unity で 3D 計算の経験も積んでいたので、是非とも自作のフレームワークを組み上げて、それを利用して 3D ゲーム制作に挑戦しようと考えました。如何にゲームエンジンに頼らずに、自作のプログラムでキャラクターを制御できるのが今回の制作の課題となります。

アピールポイント

- プレイヤーキャラクターのアクション表現、特にコンボアクションの滑らかさを重視しました
- 複雑な処理を分割し、関数化することで、保守性の向上を目指しました
- 各種関数、変数に適切な名前を付け、注釈用のコメントを書き残すことで、可読性の向上を目指しました
- ポストエフェクトシェーダー（ブルームシェーダー）の実装によって、発光や光反射の表現を試みました

クラス一覧

ゲームの基盤クラス

クラス名	説明
Game	ゲームアプリケーションの基底クラス
fpsController	ゲームの fps を制御するクラス
ActionSample	ゲームアプリケーションの本体 Game クラスを継承
SceneManager	シーン管理クラス
IScene	各シーン共通のインターフェース
SceneLoading	ローディング画面のシーンクラス
SceneTitle	タイトル画面のシーンクラス
SceneGamePlay	ゲームプレイ画面のシーンクラス
World	シーン内の各オブジェクトを管理するワールドクラス
Field	フィールドクラス
TitleCamera	タイトル画面用カメラクラス
TPCamera	三人称カメラクラス（プレイ画面用）
Light	ライトクラス
ActorGroupManager	アクターのグループを管理するクラス
ActorManager	グループ内のアクターを管理するクラス
Actor	アクターの基底クラス
Player	プレイヤーキャラクタークラス アクタークラスを継承
Enemy	敵キャラクターの基底クラス アクタークラスを継承
Ghoul	雑魚敵クラス 敵基底クラスを継承
DragonBoar	ボス敵クラス 敵基底クラスを継承

プレイヤー・敵関連クラス・構造体

クラス名	説明
Body	衝突判定基底クラス
BoundingSphere	衝突判定用球体クラス 衝突判定基底クラスを継承
BoundingCapsule	衝突判定用カプセルクラス 衝突判定基底クラスを継承
Line	衝突判定用線分クラス
Ray	衝突判定用レイクラス
PlayerInput	プレイヤーの入力判定クラス
PlayerAttack	プレイヤー攻撃の当たり判定クラス
EnemyAttack	敵攻撃の当たり判定クラス
AttackParameter	攻撃パラメータ構造体
Damage	ダメージ構造体

数学計算関連クラス・構造体

クラス名	説明
MathHelper	数学計算補助クラス
Collision	衝突計算クラス
CollisionMesh	衝突計算用メッシュクラス
CollisionTriangle	衝突計算用三角形クラス
Random	乱数生成クラス
Timer	加算式タイマークラス
CountdownTimer	カウントダウン式タイマークラス
Vector2	2 次元ベクトル構造体
Vector3	3 次元ベクトル構造体
Matrix	変換行列構造体
Quaternion	クオータニオン構造体
Color	カラー構造体

描画関連クラス

クラス名	説明
Graphics2D	2D グラフィックスクラス
Graphics3D	3D グラフィックスクラス
AnimatedSprite	アニメーション付きスプライトクラス
Billboard	ビルボードクラス
ModelAsset	モデルアセットクラス
TextureAsset	テクスチャアセットクラス
StaticMesh	スタティックメッシュクラス
SkeletalMesh	スケルタルメッシュクラス
AnimatedMesh	アニメーション付きメッシュクラス
Animation	アニメーション制御クラス

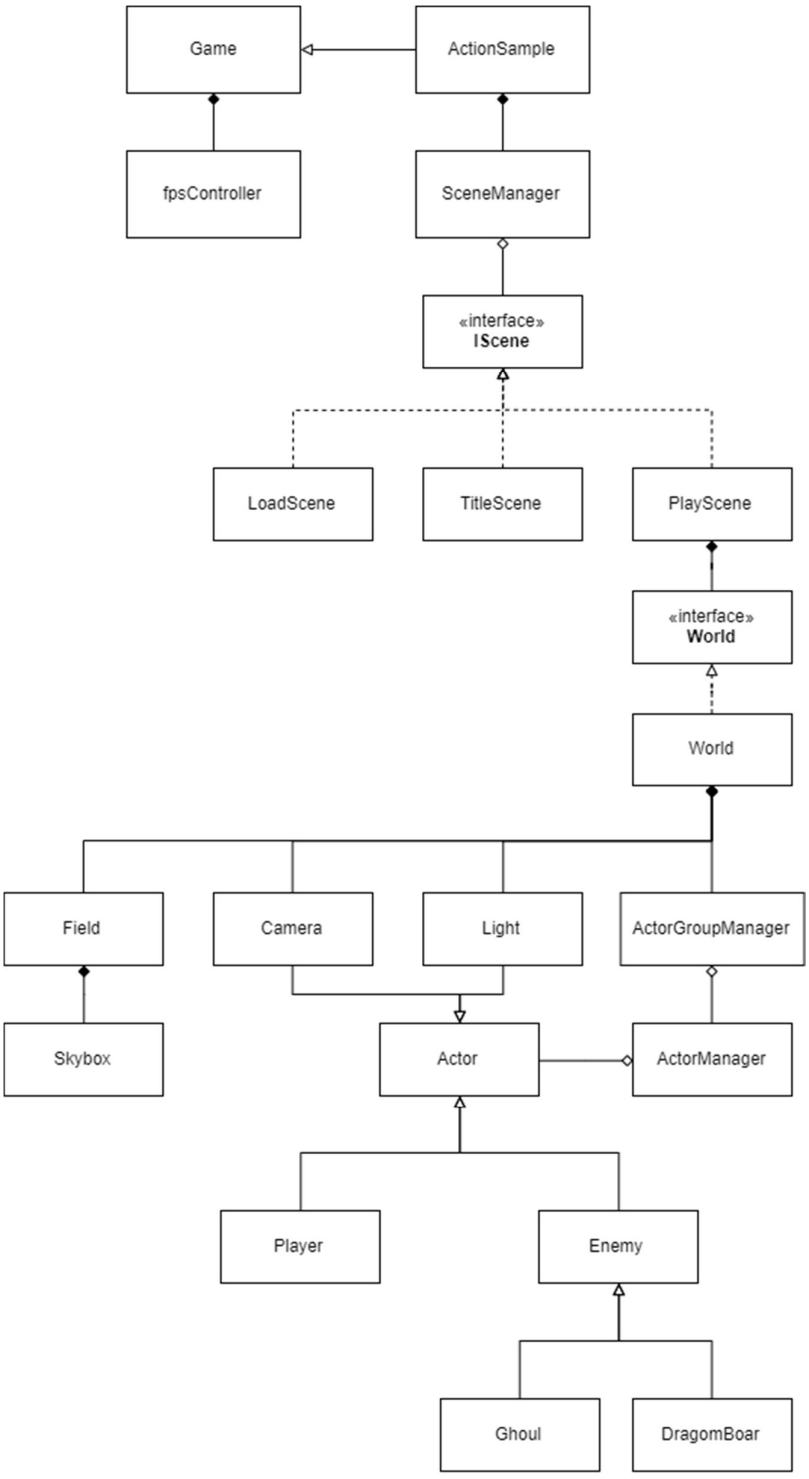
シェーダー関連クラス・構造体

クラス名	説明
Shader	シェーダークラス
ShaderManager	シェーダー管理クラス
RenderTarget	レンダーターゲットクラス
CubeMap	環境マップクラス
BloomCB	ブルームシェーダー用定数バッファ構造体
PixelShaderCB	ピクセルシェーダー用定数バッファ構造体
RadialCB	放射ブラーシェーダー用定数バッファ構造体

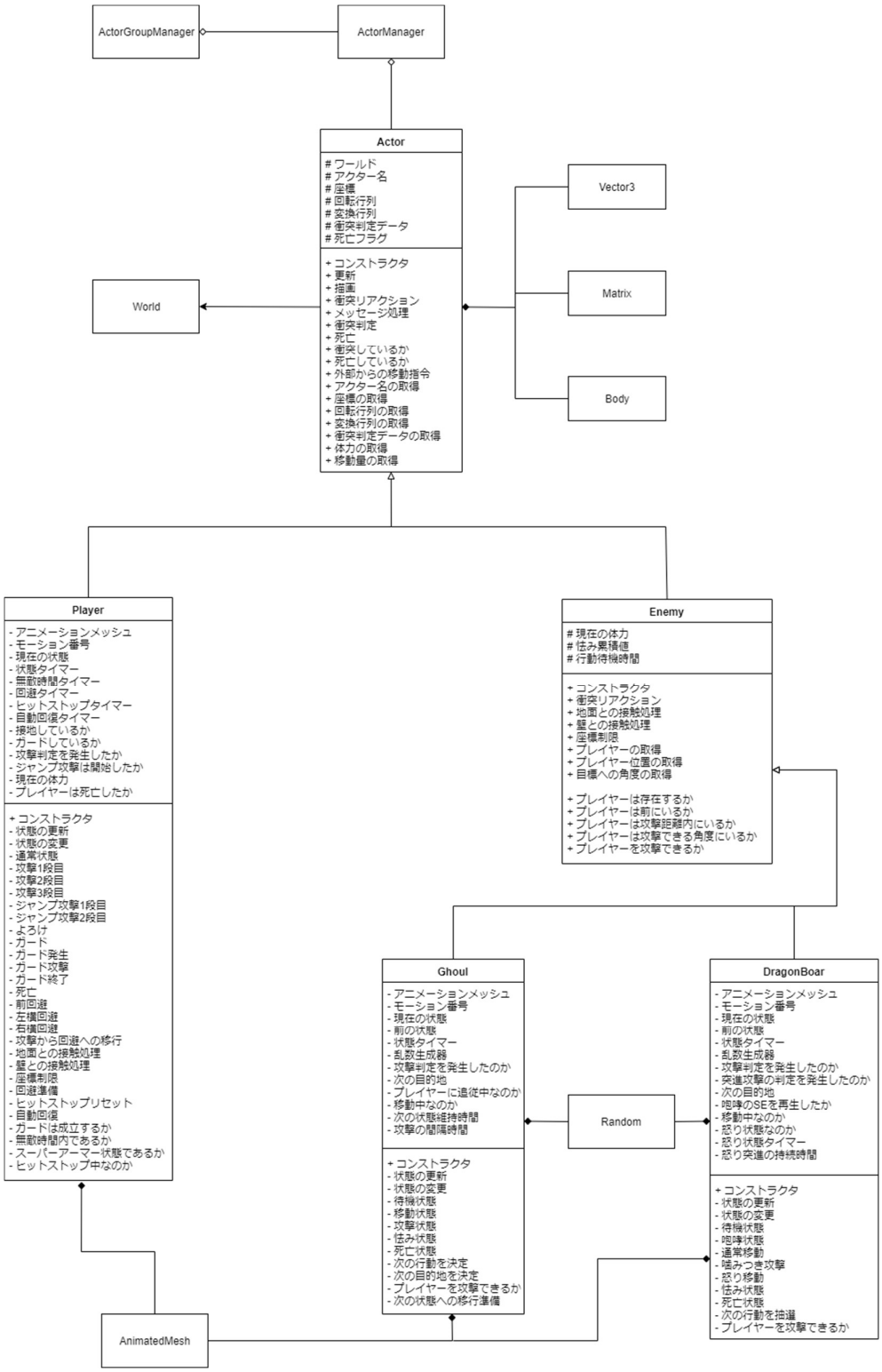
その他のクラス

クラス名	説明
GamePad	ゲームパッド入力クラス
Keyboard	キーボード入力クラス
Sound	サウンドクラス

全体のクラス図


Game
ActionSample
fpsController
SceneManager
«interface»
IScene
LoadScene
TitleScene
PlayScene
«interface»
World
World
Field
Camera
Light
ActorGroupManager
Skybox
Actor
ActorManager
Player
Enemy
Ghoul
DragomBoar

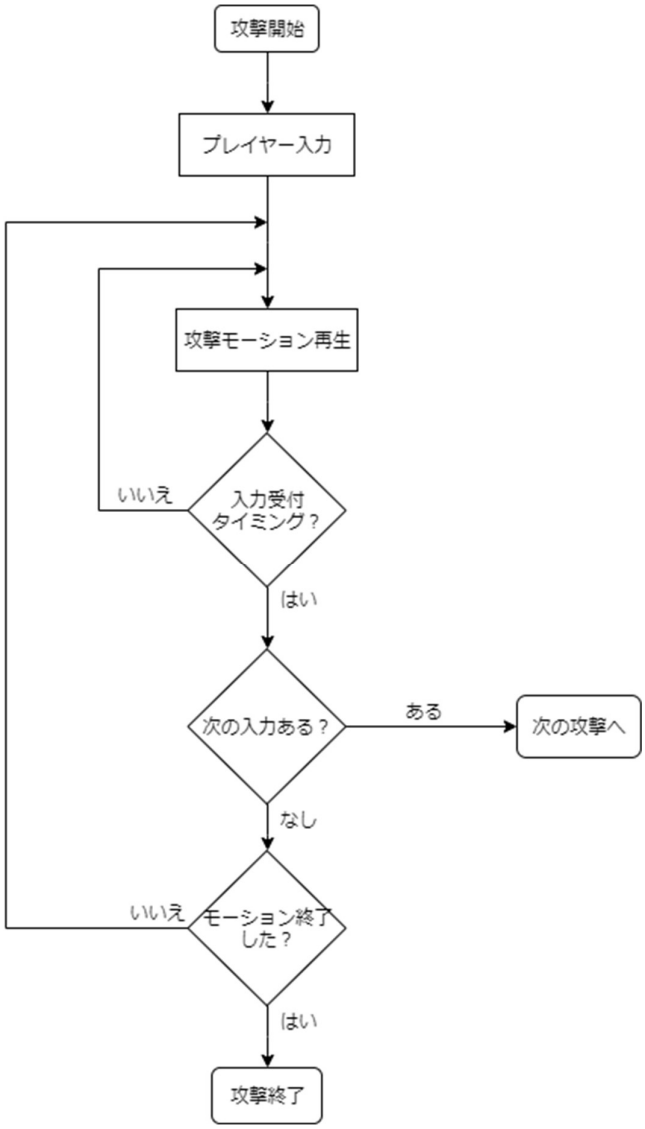
アクター関連のクラス図



プレイヤーキャラクターのモーション遷移について

- 各攻撃モーションに終了時間と次の行動への入力受付時間を設定してあり、入力受付中にボタンが押されると、次の行動に移行します
- 入力判定はプレイヤー入力判定クラスに行っています。またモーション終了タイミングと入力受付開始・終了タイミングは別々で設定してあるため、ディレイ入力にも対応します
- 入力タイミングとモーション再生速度・終了タイミングなどは全部手作業で調整しています

以下は連続攻撃の流れ：



敵キャラクターの設計について

今回の作品には、雑魚敵（Ghoul）とボス敵（DragonBoar）が存在します。両者の仕様は以下となります。

行動選択の基本仕様：

- 次に行う行動は基本、乱数で決めます
- 待機や移動状態の維持時間は乱数でブレを付けます（例：雑魚敵の移動状態の維持時間は６～８秒）
- 待機状態は連続に発生しません

移動状態の基本仕様：

- プレイヤーの位置を検出し、プレイヤーを追従して移動します
- プレイヤーがリーチ内にいれば、少しだけ間を空けて攻撃状態に移行します

雑魚敵（Ghoul）の仕様：

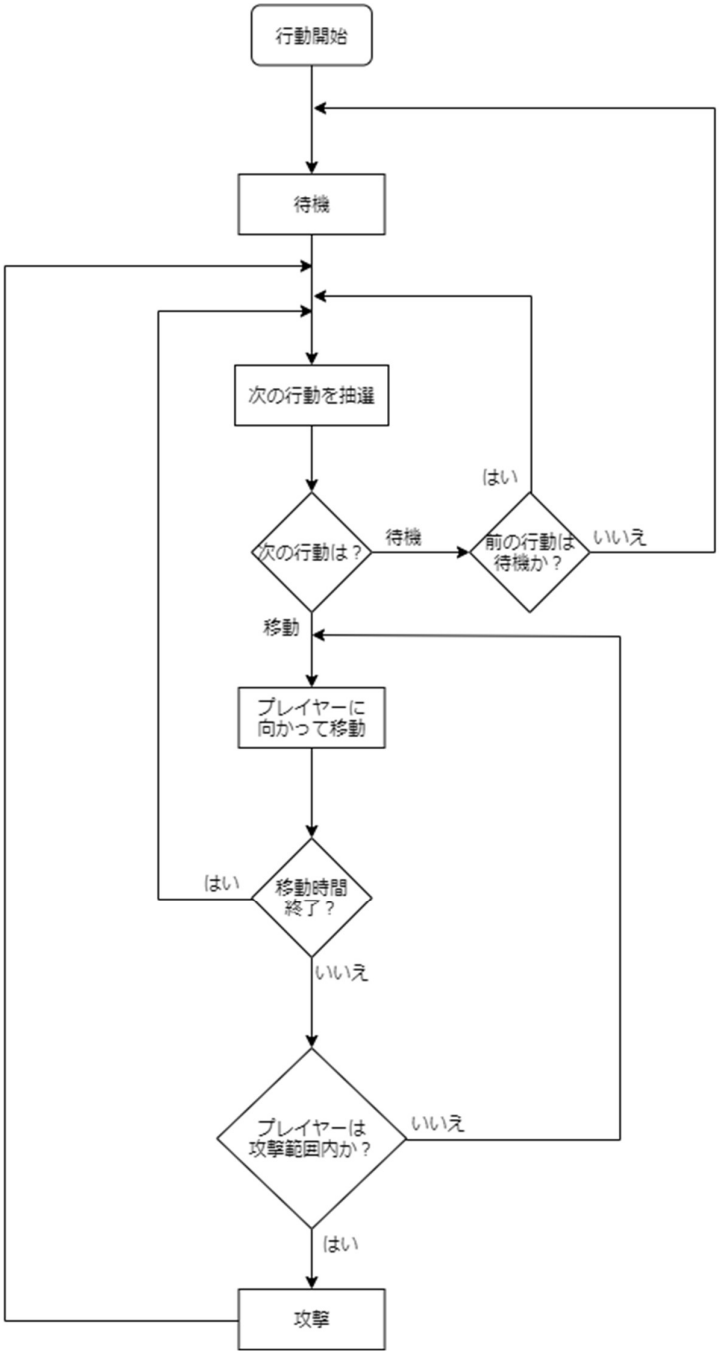
- プレイヤーの攻撃を受けると、ダメージ量、自身の状態を問わずに怯みます
- 体力量は少ないので（約 5 回攻撃で倒せる)、画面上に体力ゲージを表示していません

ボス敵（DragonBoar）の仕様：

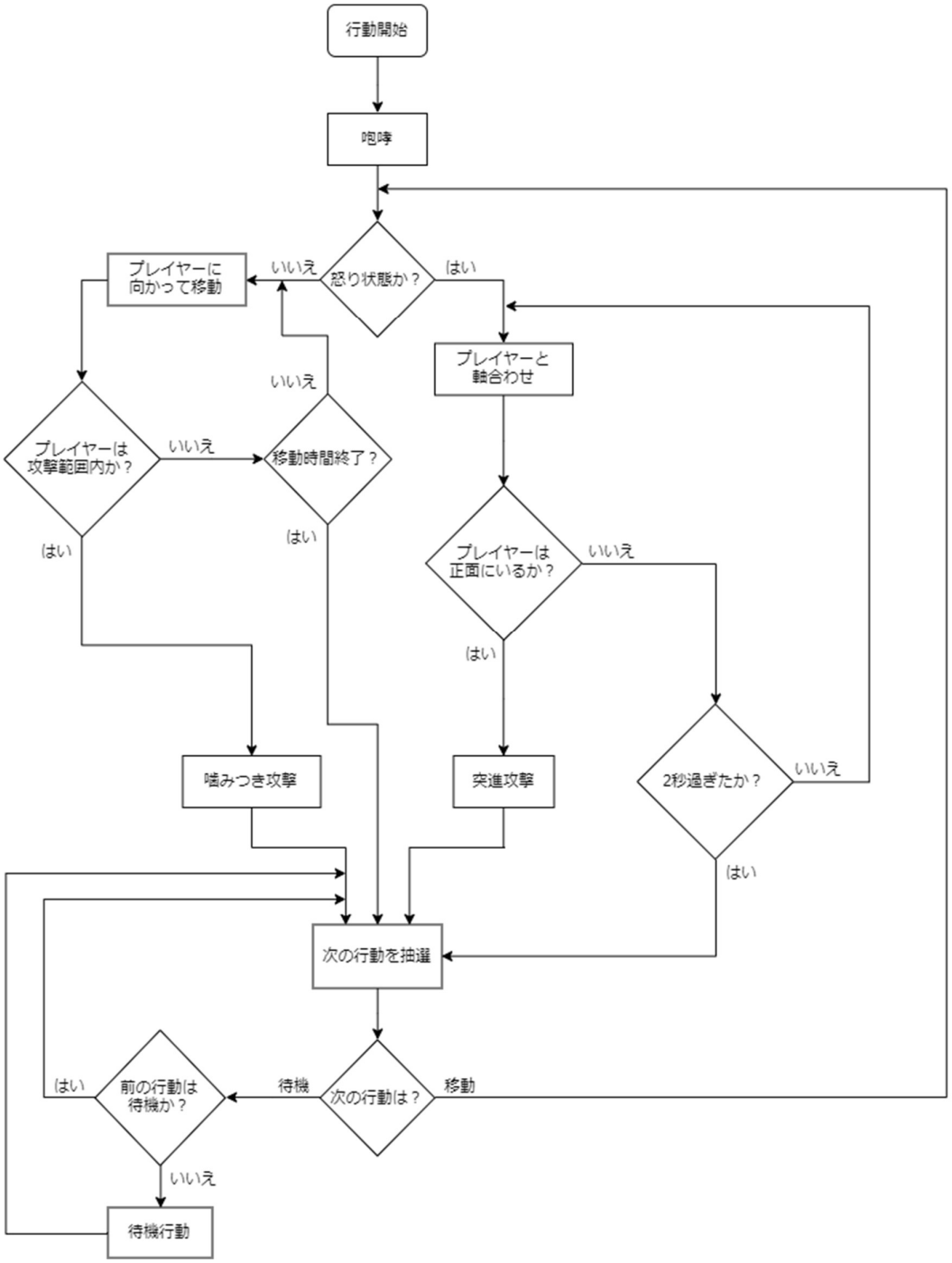
- 体力量が多くので、画面の下に体力ゲージを表示しています
- 体力を半分削ると、2 分間怒り状態になります（攻撃パターンが変わります）

敵キャラクターの行動ルーチンのフローチャート：

- 雑魚敵（Ghoul）



- ボス敵（DragonBoar）



良かったと思うところ

- プレイヤーキャラクターのモーション遷移は滑らかで、市販のゲーム顔負けの水準になっています
- できるだけ可読性を高めて、仕様の追加、変更やデバッグはやりやすいです
- ゲームの進行に影響を及ぼすバグもなく、アプリケーションとしては安定しています
- 製作の間に修得した技術を作品に取り込んでいき、ステップアップしていることを実感できました
- ゲームエンジンに頼らずに作品を完成できて、いい経験になりました

これから挑戦したいこと

- より凝った敵の行動パターンに挑戦したいと思います。
  - 更に言うと、ゲーム AI の制作に興味あります。
- 今回はオブジェクト指向でプログラムの設計を行いましたが、最近コンポーネント指向という概念を学んだので、今度はそれでプログラム設計を行ってみたいと思います。
- シェーダープログラミングを精進し、より高度な映像表現に挑戦したいと思います。