システム制御レポート 3

23B30925 横井風羽

2025/12/13

1　静的構造

1.1　モデリングしたクラス図

Packman ゲームのクラス図を以下に示す。

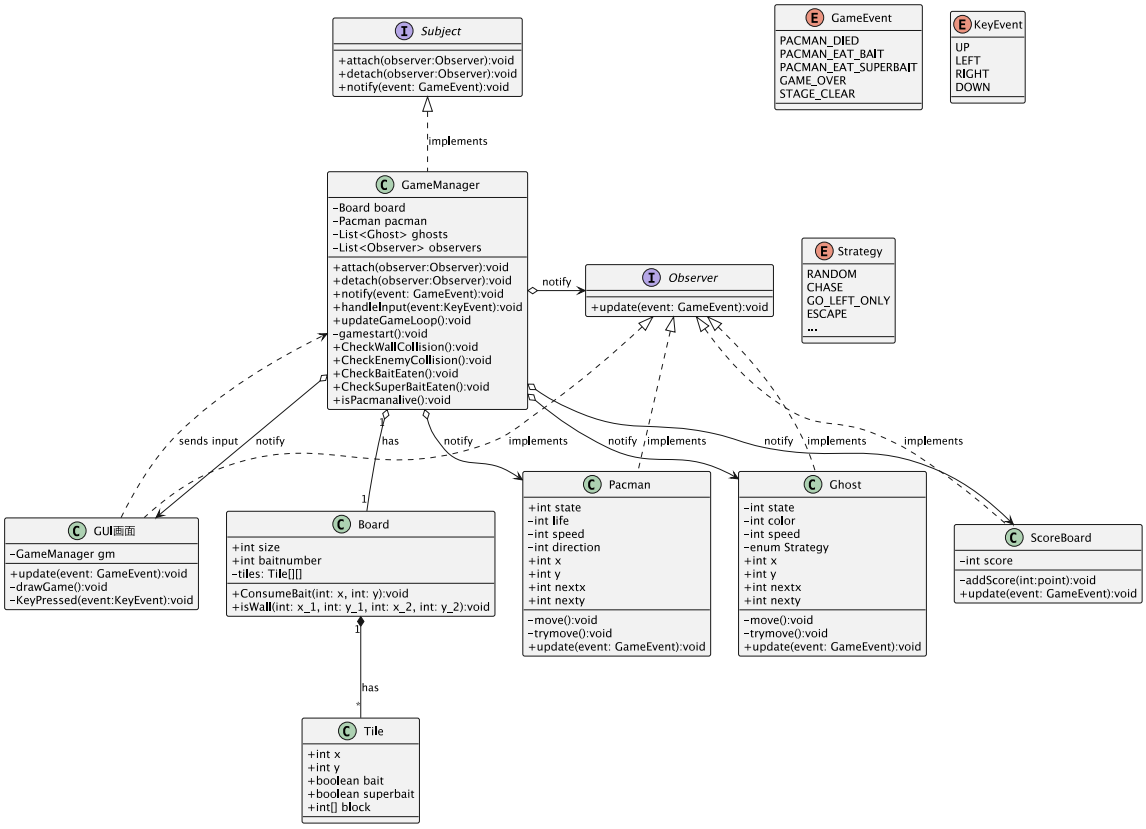

図 1　クラス図

1.2 主要なクラスの簡単な説明

- **GameManager (GM)**: ゲーム全体の状態（パックマン、ゴースト、ボードなど）を保持し、ゲームループや入力処理、衝突判定などのロジックを統括する。Subject インターフェースを実装し、Observer へイベントを通知する。
- **Pacman**: プレイヤーが操作するキャラクター。自身の位置、ライフ、速度、移動方向などを管理する。Observer として GameManager からのイベントを受け取り、状態を更新する。
- **Ghost**: 敵キャラクター。移動戦略（Strategy パターンを想定）に基づき自動で動作する。Pacman と同様に Observer として機能する。
- **Board**: ゲーム盤面を表す。タイルの配置、壁の判定、エサの管理を行う。
- **GUI 画面 (Display)**: ゲームの視覚的表現を担当する。Observer としてイベントを受け取り、画面の再描画を行う。
- **Subject / Observer**: Observer パターンを実現するためのインターフェース。イベント駆動型の更新通知を行う。

1.3 全体の設計理由

ゲームの状態変化（エサを食べた、衝突した、クリアした等）を複数のコンポーネント（画面表示、スコア、各キャラクター）に効率よく伝達するために、**Observer パターン**を採用した。これにより、`GameManager` がイベントを発行するだけで、各オブジェクトが自律的に対応する動作を行えるようになり、結合度を下げている。また、ゴーストの動きには将来的に様々なアルゴリズム（ランダム、追跡、先回り等）を適用できるよう配慮した設計としている。

2 動的振舞いの例

2.1 シナリオ実行のシーケンス図

以下のシナリオにおけるシーケンス図を示す。

1. パックマンを右に移動させエサを獲得する。
2. もう 1 マス右に行こうとしたら壁があって動けない。
3. その後、モンスターがぶつかってきてライフが 1 つ減り、リスタートする。

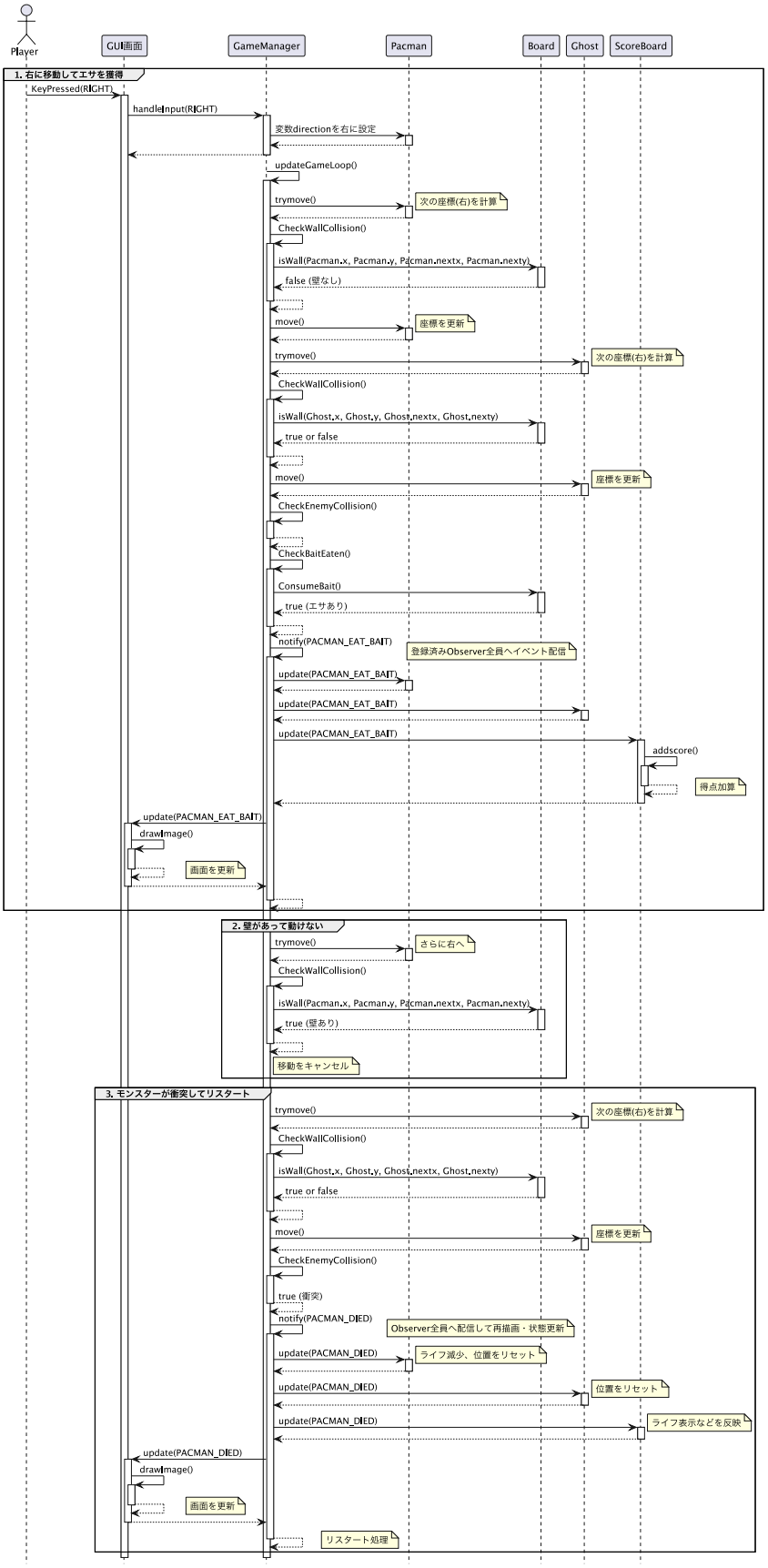
Player
GUI画面
GameManager
Pacman
Board
Ghost
ScoreBoard
1. 右に移動してエサを獲得
KeyPressed(RIGHT)
handleInput(RIGHT)
変数directionを右に設定
updateGameLoop()
trymove()
次の座標(右)を計算
CheckWallCollision()
isWall(Pacman.x, Pacman.y, Pacman.nextx, Pacman.nexty)
false (壁なし)
move()
座標を更新
trymove()
次の座標(右)を計算
CheckWallCollision()
isWall(Ghost.x, Ghost.y, Ghost.nextx, Ghost.nexty)
true or false
move()
座標を更新
CheckEnemyCollision()
CheckBaitEaten()
ConsumeBait()
true (エサあり)
notify(PACMAN_EAT_BAIT)
登録済みObserver全員へイベント配信
update(PACMAN_EAT_BAIT)
update(PACMAN_EAT_BAIT)
update(PACMAN_EAT_BAIT)
addscore()
得点加算
update(PACMAN_EAT_BAIT)
drawImage()
画面を更新
2. 壁があって動けない
trymove()
さらに右へ
CheckWallCollision()
isWall(Pacman.x, Pacman.y, Pacman.nextx, Pacman.nexty)
true (壁あり)
移動をキャンセル
3. モンスターが衝突してリスタート
trymove()
次の座標(右)を計算
CheckWallCollision()
isWall(Ghost.x, Ghost.y, Ghost.nextx, Ghost.nexty)
true or false
move()
座標を更新
CheckEnemyCollision()
true (衝突)
notify(PACMAN_DIED)
Observer全員へ配信して再描画・状態更新
update(PACMAN_DIED)
ライフ減少、位置をリセット
update(PACMAN_DIED)
位置をリセット
update(PACMAN_DIED)
ライフ表示などを反映
update(PACMAN_DIED)
drawImage()
画面を更新
リスタート処理

図２　シ3ケンス図

このシーケンス図では、ユーザー入力 KeyPressed から始まり、GameManager が移動判定 trymove、壁判定 CheckWallCollision、イベント通知 notify を行う流れを表現している。特に、エサ獲得時（PACMAN_EAT_BAIT）や衝突時（PACMAN_DIED）に、Observer パターンを通じて各インスタンスが一斉に更新される様子が確認できる。

3 独自に想定した仮定

なし

4 工夫点

- **Observer パターンの活用**: ゲーム内のイベント（死亡、スコア獲得など）を一元管理し、スパゲッティコード化を防いでいる点。
- **責任の分離**: Board クラスに盤面判定ロジックを集約し、キャラクタークラスは自身の移動ロジックに集中できるようにした点。
- **拡張性**: ゴーストの行動パターンを Strategy として定義可能な構造にしており、後からの機能追加を容易にしている点。

5 感想

クラス図とシーケンス図を作成することで、実装前にオブジェクト間の責務分担を明確にすることの重要性を再認識した。特に Observer パターンを導入することで、イベント処理の流れが整理され、見通しの良い設計になったと感じる。PlantUML を用いることで、修正が容易なテキストベースでのモデリングができ、試行錯誤がスムーズに行えた。