中間レポート　22B30098　一瀬達矢

1. モデリングしたクラス図

ダイアグラム, 設計図

AI によって生成されたコンテンツは間違っている可能性があります。

1. 設計したクラス群が以下のシナリオを実行するシーケンス図 • 「パックマンを右に移動させエサを獲得したが， もう１マス右に行こうとしたら壁があって動けず モンスターがぶつかってきてライフが1つ減ってリスタート」

ダイアグラム, 概略図

AI によって生成されたコンテンツは間違っている可能性があります。

定期的な処理

ダイアグラム

AI によって生成されたコンテンツは間違っている可能性があります。

1. 独自に想定した仮定（なければ「なし」とだけ記述）

なし

1. 主要なクラスの簡単な説明．設計理由が含まれると好ましい

全体的に構造に関してはCompositeパターン、振る舞いに関してはObserverパターンを採用した。

以下の構成はCompositeパターンを設計理由としている。

Gameクラスは以下の主要な4つのフィールドを持つ。

Pacman: パックマンの動作に関するメソッドを持つ

Map : ゲーム内のオブジェクト全般を管理する

Timer: ゲームの状態を管理する

さらに、Map,Timerを具体的に述べると、

Mapはエサや壁、ゴーストをフィールドに持ち、それぞれに関するメソッドを持つ。エサと壁についてはGameObjectというインターフェイスを実現している。

ゴーストについてはゴーストごとに戦略が変わるので、抽象クラスで定義しゴーストの特色ごとにchase()というメソッドを変えている。

TimerはObserverパターンを参考にし、通常のタイマー機能に加えて、定期的に

「ゴーストと衝突したかどうか」「ゲームオーバーかどうか」などを判定し、これらに該当する場合はnotifyに相当するtrigger{トリガー名}というGameクラスのメソッドを呼ぶことで、ゲームの状態も管理している。

(trigger{トリガー名}は例えば、triggerTimeOut()など)

動作責務は主にGameが持ち、

判定責務は「ゴーストと接触したかどうか」「タイムアウトかどうか」をTimerが、「エサを全て食べたか」「パワーエサを食べたか」「パックマンが壁と衝突したか」などのPacmanを動かさない限り発生しないような状況の判定はGameが行っている。

工夫点

主要なクラスはGameクラスが全てを管理している、GUIはGameクラスにアクセスするだけで良く、その下でどのクラスが実際に実行しているかは気にしなくて良い

また、クラス図を見たときにゲームの構成が分かりやすいように、木構造を形成している。

(Compositeパターン)

加えて、Timerというクラスにより定期的に「ゴーストと接触したかどうか」「タイムアウトかどうか」などを一括で管理することで可読性が向上している。

1. 感想

パックマンのゲームほどシンプルなものでも様々なモデリングが考えられることを実感した。実際のソフトウェア開発ではさらに複雑になると思うので、開発前にモデリングを議論することは極めて有意義なことだと思った。

今回のソースコードについては以下のGithub上に公開しています。